# Pre- and post-quantum Diffie-Hellman from groups, actions, and isogenies

Benjamin Smith

# Pre- and post-quantum Diffie–Hellman from groups, actions, and isogenies

Benjamin Smith

Inria *and* Laboratoire d'Informatique de l'École polytechnique (LIX),
Université Paris–Saclay, France
smith@lix.polytechnique.fr

**Abstract.** Diffie–Hellman key exchange is at the foundations of public-key cryptography, but conventional group-based Diffie–Hellman is vulnerable to Shor's quantum algorithm. A range of "post-quantum Diffie–Hellman" protocols have been proposed to mitigate this threat, including the Couveignes, Rostovtsev–Stolbunov, SIDH, and CSIDH schemes, all based on the combinatorial and number-theoretic structures formed by isogenies of elliptic curves. Pre- and post-quantum Diffie–Hellman schemes resemble each other at the highest level, but the further down we dive, the more differences emerge—differences that are critical when we use Diffie–Hellman as a basic component in more complicated constructions. In this survey we compare and contrast pre- and post-quantum Diffie–Hellman algorithms, highlighting some important subtleties.

## 1 Introduction

The Diffie–Hellman key-exchange protocol is, both literally and figuratively, at the foundation of public-key cryptography. The goal is for two parties, Alice and Bob, to derive a shared secret from each other's public keys and their own private keys. Diffie and Hellman's original solution [52] is beautifully and brutally simple: given a fixed prime $p$ and a primitive element $g$ in the finite field $\mathbb{F}_p$ (that is, a generator of the multiplicative group $\mathbb{F}_p^\times$), Alice and Bob choose secret keys $a$ and $b$, respectively, in $\mathbb{Z}/(p-1)\mathbb{Z}$. Alice computes and publishes her public key $A = g^a$, and Bob his public key $B = g^b$; the shared secret value is $S = g^{ab}$, which Alice computes as $S = B^a$, Bob as $S = A^b$.

The security of the shared secret depends on the hardness of the *Computational Diffie–Hellman Problem* (CDHP), which is to compute $S$ given only $A$, $B$, and the public data of the structures that they belong to. For finite-field Diffie–Hellman, this means computing $g^{ab}$ given only $g$, $g^a$, and $g^b \pmod p$. The principal approach to solving the CDHP is to solve the *Discrete Logarithm Problem* (DLP), which is to compute $x$ from $g$ and $g^x$. We thus recover $a$ from $A = g^a$ (or, equivalently, $b$ from $B = g^b$), then power $B$ by $a$ (or $A$ by $b$) to recover $S$. Attacking the DLP means directly attacking one of the public keys, regardless of any particular shared secret they may be used to derive.

Over the past four decades, the Diffie–Hellman protocol has been generalized from multiplicative groups of finite fields to a range of other algebraic groups,

most notably elliptic curves. Partly motivated by this cryptographic application, there has been great progress in discrete logarithm algorithms for some groups, most notably Barbulescu, Gaudry, Joux, and Thomé's quasipolynomial algorithm for discrete logarithms in finite fields of fixed tiny characteristic [11].

The most stunning development in discrete logarithm algorithms came with the rise of the quantum computation paradigm: Shor's quantum algorithm [124] solves the discrete logarithm problem—and thus breaks Diffie–Hellman—in any group in polynomial time and space on a quantum computer.[1] The development of quantum computers of even modest capacity capable of running Shor's algorithm remains an epic challenge in experimental physics: at the time of writing, the largest implementation of Shor's algorithm was used to factor the integer 21, so there is some way to go [96]. But in anticipation, cryptographic research has already bent itself to the construction of *post-quantum* cryptosystems, designed to be used on conventional computers while resisting known quantum attacks.

Nowadays, Diffie–Hellman is often an elementary component of a more complicated protocol, rather than the entire protocol itself. For example, the TLS protocol used to establish secure internet connections includes an ephemeral Diffie–Hellman [117]. But to give a more interesting example, the X3DH protocol [95] used to establish connections in Signal and WhatsApp includes *four* simple Diffie–Hellmans between various short and long-term keypairs. The common use of Diffie–Hellman as a component makes the search for a drop-in post-quantum replacement for classical Diffie–Hellman particularly relevant today.

While many promising post-quantum candidates for public-key encryption and signatures have been developed—the first round of the NIST post-quantum standardization process [108] saw 59 encryption/KEM schemes and 23 signature schemes submitted—finding a simple post-quantum drop-in replacement for Diffie–Hellman (as opposed to a KEM) has proven to be surprisingly complicated. Some interesting post-quantum "noisy Diffie–Hellman" key exchange protocols based on hard problems in codes, lattices, and Ring-LWE have been put forward over the years (including [4], [53], [54], [110], [6], [23], and [50]), but these typically require a reconciliation phase to ensure that Alice and Bob have the same shared secret value (as opposed to an approximate shared secret with acceptable noise on each side); we will not discuss these protocols further here. Perhaps surprisingly, given the loudly trumpeted quantum destruction of elliptic curve cryptography by Shor's algorithm, the most serious candidates for post-quantum Diffie–Hellman come from isogeny-based cryptography, which is founded in the deeper theory of elliptic curves.

The key idea in moving from conventional elliptic-curve cryptography to isogeny-based cryptography is that points on curves are replaced with entire curves, and relationships between points (scalars and discrete logarithms) are replaced with relationships between curves (isogenies). Isogeny classes have just

[1] More generally, Armknecht, Gagliardoni, Katzenbeisser, and Peter have shown that no group-homomorphic cryptosystem can be secure against a quantum adversary, essentially because of the existence of Shor's algorithm [8].

enough algebraic structure to define efficient asymmetric cryptosystems, but not enough to make them vulnerable to Shor's algorithm.

But what should a "post-quantum Diffie–Hellman" scheme be, and how closely should it match classical Diffie–Hellman functionalities and semantics? To what extent can the intuition and theoretical lore built up over decades of classical Diffie–Hellman carry over to these new protocols? This survey is an attempt to begin addressing these questions. The aim is to help cryptographers, mathematicians, and computer scientists to understand the similarities and differences between classical Diffie–Hellman and the new post-quantum protocols.

*The plan.* We begin with a quick survey of classical group-based Diffie–Hellman in §§2-5. We discuss modern elliptic-curve Diffie–Hellman in §7; this dispenses with the underlying group on some levels, and thus forms a pivot for moving towards post-quantum Diffie–Hellman. We review Couveignes' *hard homogeneous spaces* framework in §8 and §9, before introducing HHS cryptosystems in the abstract in §10; we go deeper into the underlying hard problems in §11 and §12. Moving into the concrete, we recall basic facts about isogenies in §13, before discussing commutative isogeny-based key exchange in §14 and the stranger SIDH scheme in §15. Our focus is mostly constructive, and our discussion of quantum cryptanalysis will be purely asymptotic, for reasons discussed in §6.

*Limiting scope.* The basic Diffie–Hellman scheme is completely unauthenticated: it is obviously vulnerable to a man-in-the-middle attack where Eve impersonates Bob to Alice, and Alice to Bob. Alice and Bob must therefore authenticate each other outside the Diffie–Hellman protocol, but we do not discuss authentication mechanisms here. We also ignore the provable-security aspects of these protocols, beyond noting that each has been proven session-key secure in Canetti and Krawczyk's adversarial authenticated-links model [32] (see [46, §5.3] for a proof for commutative isogeny key exchange, and [45, §6] for SIDH). As we noted above, we do not discuss noisy Diffie–Hellman schemes here, mostly for lack of time and space, but also because these are further from being drop-in replacements for classical Diffie–Hellman. Finally, we must pass over *decision* Diffie–Hellman-based protocols in silence, partly for lack of space, but mostly because at this early stage it seems hard to say anything nontrivial about decision Diffie–Hellman in the post-quantum setting. We do this with some reluctance: as Boneh declared in [19], "the decision Diffie–Hellman assumption is a gold mine" (at least for theoretical cryptographers). Revisiting [19] in the post-quantum setting would be highly interesting, but this is neither the time nor the place for that investigation.

*Notation.* We will use abelian groups written additively and multiplicatively, depending on the context. To minimise confusion, we adopt these typographical conventions for groups and elements:

- $\mathcal{G}$ always denotes an abelian group written **additively**, with group operation $(P, Q) \mapsto P + Q$, inverse $P \mapsto -P$, and identity element 0.

– $\mathfrak{G}$ always denotes an abelian group written **multiplicatively**, with group operation $(\mathfrak{p}, \mathfrak{q}) \mapsto \mathfrak{pq}$, inverse $\mathfrak{p} \mapsto \mathfrak{p}^{-1}$, and identity element 1.

*Acknowledgements.* I am grateful to Luca De Feo, Florian Hess, Jean Kieffer, and Antonin Leroux for the many hours they spent discussing these cryptosystems with me, and the organisers, chairs, and community of WAIFI 2018.

## 2 Abstract groups and discrete logarithms

Diffie and Hellman presented their key exchange in the multiplicative group of a finite field, but nothing in their protocol requires the field structure. We will restate the protocol in the setting of a general finite abelian group in §3; but first, we recall some basic facts about abstract groups and discrete logarithms.

Let $\mathcal{G}$ be a finite abelian group of order $N$ (written additively, following the convention above). We can assume $\mathcal{G}$ is cyclic. For every integer $m \pmod N$ we have an exponentiation endomorphism $[m] : \mathcal{G} \to \mathcal{G}$, called *scalar multiplication*, defined for non-negative $m$ by

$$[m] : P \longmapsto \underbrace{P + \cdots + P}_{m \text{ copies}}$$

and for negative $m$ by $[m]P = [-m](-P)$. We can compute $[m]$ in $O(\log m)$ $\mathcal{G}$-operations using a variety of addition chains; typically $m \sim \#\mathcal{G} = N$.

The fundamental hard algorithmic problem in $\mathcal{G}$ is computing the inverse of the scalar multiplication operation: that is, computing discrete logarithms.

**Definition 1 (DLP).** *The* Discrete Logarithm Problem *in $\mathcal{G}$ is, given $P$ and $Q$ in $\langle P \rangle \subseteq \mathcal{G}$, compute an $x$ such that $Q = [x]P$.*

Any DLP instance in any $\mathcal{G}$ can always be solved using $O(\sqrt{N})$ operations in $\mathcal{G}$, using (for example) Shanks' baby-step giant-step algorithm (BSGS), which also requires $O(\sqrt{N})$ space [123]; Pollard's $\rho$ algorithm reduces the space requirement to $O(1)$ [114]. If $N$ is composite and its (partial) factorization is known, then we can do better using the Pohlig–Hellman–Silver algorithm [113], which solves the DLP by reducing to the DLP in subgroups of $\mathcal{G}$ (see §12 below).

The DLP enjoys random self-reducibility: if we have an algorithm that solves DLPs for a large fraction $1/M$ of all possible inputs, then we can solve DLPs for all possible inputs after an expected $M$ random attempts. Suppose we want to solve an arbitrary DLP instance $Q = [x]P$. We choose a random integer $r$, try to solve $Q' = Q + [r]P = [x + r]P$ for $x + r$, and if we succeed then we recover $x = (x + r) - r$. After $M$ randomizations, we expect to find an $r$ for which $Q'$ lands in the set of inputs to which the algorithm applies.

In the pure abstract, we consider $\mathcal{G}$ as a black-box group: operations are performed by oracles, and elements are identified by (essentially random) bit-strings. This models the absence of useful information that we could derive from any concrete representation of $\mathcal{G}$. In this setting, Shoup [125] has proven that

the complexity of solving the DLP is not merely in $O(\sqrt{N})$, but in $\Theta(\sqrt{N})$. But in the real world, we do not have black-box groups; every group has a specific concrete element representation and an explicit algorithmic group law. The difficulty of the DLP then varies with the representation, as we will see in §5.

## 3 Pre-quantum Diffie–Hellman

Now let us consider Diffie–Hellman in the abstract. Let $\mathcal{G}$ be a cyclic group, and fix a public generator $P$ of $\mathcal{G}$. Public keys are elements of $\mathcal{G}$; private keys are bitstrings, interpreted as elements of $\mathbb{Z}/N\mathbb{Z}$. Each (public,private)-keypair $(Q = [x]P, x)$ presents a DLP instance in $\mathcal{G}$.

The Diffie–Hellman protocol takes place in into two logical phases, which in practice may be separated by a significant period of time. In the first phase, the parties generate their keypairs using Algorithm 1 (`KeyPair`):

- Alice generates her keypair as $(A, a) \leftarrow$ `KeyPair()` and publishes $A$;
- Bob generates his as $(B, b) \leftarrow$ `KeyPair()` and publishes $B$.

In the second phase, they compute the shared secret $S$ with Algorithm 2 (`DH`):

- Alice computes $S \leftarrow \texttt{DH}(B, a)$;
- Bob computes $S \leftarrow \texttt{DH}(A, b)$.

Alice and Bob have the same value $S$ because $S = [a]B = [b]A = [ab]P$.

**Algorithm 1:** Keypair generation for textbook Diffie–Hellman in a group $\mathcal{G} = \langle P \rangle$ of order $N$.

**Input:** ()
**Output:** A pair $(Q, x)$ in $\mathcal{G} \times \mathbb{Z}/N\mathbb{Z}$ such that $Q = [x]P$

```
function KeyPair()
    x ←$ Z/NZ
    Q ← [x]P
    return (Q, x)
```

**Algorithm 2:** Textbook Diffie–Hellman key exchange in $\mathcal{G}$

**Input:** A public key $R$ in $\mathcal{G}$, and a private key $x$ in $\mathbb{Z}/N\mathbb{Z}$
**Output:** A shared secret $S \in \mathcal{G}$

```
function DH(R, x)
    S ← [x]R
    return S                          // To be input to a KDF
```

The security of the (entire) shared secret depends on the hardness of the *Computational Diffie–Hellman Problem* (CDHP) in $\mathcal{G}$.

**Definition 2 (CDHP).** *The* Computational Diffie–Hellman Problem *in* $\mathcal{G}$ *is, given* $P$, $A = [a]P$, *and* $B = [b]P$ *in* $\mathcal{G}$, *to compute* $S = [ab]P$.

While it is obvious that an algorithm that solves the DLP in $\mathcal{G}$ can solve the CDHP in $\mathcal{G}$, constructing a reduction in the other direction—that is, efficiently solving DLP instances given access to an oracle solving CDHP instances—is a much more subtle matter. It is now generally believed, following the work of den Boer [49], Maurer and Wolf [97,98], Muzereau, Smart, and Vercauteren [106], Bentahar [13], and Boneh and Lipton [20] that the DLP and CDHP are equivalent for the kinds of $\mathcal{G}$ that cryptographers use in practice.[2] Since solving DLP instances is the only way we know to solve CDHP instances, Diffie–Hellman is generally considered to be a member of the DLP-based family of cryptosystems.

The shared secret $S$ is *not* suitable for use as a key for symmetric cryptosystems[3]; rather, it should be treated with a Key Derivation Function (KDF) to produce a proper symmetric key $K$. This essentially hashes the secret $S$, spreading the entropy of $S$ uniformly throughout $K$, so deriving any information about $K$ requires computing the whole of $S$, hence solving a CDHP in $\mathcal{G}$. The indistinguishability of $S$, and hence its security as a key, depends on the weaker *Decisional* Diffie–Hellman Problem, which is beyond the scope of this article.

The lifespan of keypairs is crucial in Diffie–Hellman-based cryptosystems. In *ephemeral* Diffie–Hellman, Alice and Bob's keypairs are unique to each execution of the protocol. Ephemeral Diffie–Hellman is therefore essentially interactive. In contrast, *static* Diffie–Hellman uses long-term keypairs across many sessions. Alice may obtain Bob's long-term public key and complete a Diffie–Hellman key exchange with him—and start using the shared secret—without any active involvement on his part. Static Diffie–Hellman is therefore an important example of a *Non-Interactive Key Exchange* (NIKE) protocol [62].

Efficient *public-key validation*—that is, checking that a public key was honestly generated—is an important, if often overlooked, requirement for many Diffie–Hellman systems, particularly those where keys are re-used. Suppose Alice

[2] Suppose $N$ is prime. The Maurer reduction for groups of order $N$ requires an auxiliary elliptic curve over $\mathbb{F}_N$ whose order is $B$-smooth—that is, such that every prime dividing the order of the auxiliary curve is less than $B$—for some small $B$ which determines the efficiency of the reduction. If we require polynomially small $B$, then we get a polynomial-time reduction; but the hypothesis that such curves exist and can be efficiently constructed for arbitrary $N$ is extremely strong, if not wildly overoptimistic. If $B$ is simply smaller than $N^{1/2}$, then we get a reduction that is dominated by the cost of an equivalent DLP calculation, which is better than nothing; it is not so hard to construct such curves (as Bentahar does). The middle ground is to accept subexponential $B$, and hence a subexponential reduction, as Muzereau, Smart, and Vercauteren do. Brown [28] takes a more constructive approach, constructing cryptographic elliptic curves equipped with a polynomially smooth auxiliary curve.

[3] Some protocols do use the shared secret $S$ as a key, most notably the textbook ElGamal encryption presented at the start of §4.

derives a shared secret key $K$ from a Diffie–Hellman exchange with Bob's public key $B$, and then uses $K$ to communicate with Bob. A malicious Bob might construct an invalid public key $B$ in such a way that $K$ reveals information about Alice's secret key $a$. If $(a, A)$ is ephemeral then Bob has learned nothing useful about $a$, since it will never be used again; but the keypair $(A, a)$ is to be reused, as in static Diffie–Hellman, then secret information has been leaked, and Alice thus becomes vulnerable to active attacks (see e.g. [93] for an example). Public key validation is simple in a finite field: it usually suffices to check the order of the element. Antipa, Brown, Menezes, Struik, and Vanstone describe the process for elliptic-curve public keys [7], and their methods extended to most curve-based algebraic groups without serious difficulty. We will see that this is a more serious problem in post-quantum systems.

## 4 Encryption and key encapsulation

The classic ElGamal public-key encryption scheme [58] is closely related to Diffie–Hellman key exchange. Its key feature is that messages are viewed as elements of the group $\mathcal{G}$, so adding a random-looking element of $\mathcal{G}$ to a message in $\mathcal{G}$ acts as encryption.

Algorithm 3 lets Alice encrypt a message to Bob. Alice first generates an ephemeral keypair $(E, e)$, completes the Diffie–Hellman on her side using Bob's static public key $B$ to compute a shared secret $S$, which she uses as a (symmetric) key to encrypt an element of $\mathcal{G}$ via $M \mapsto C = M + S$ (with corresponding decryption $C \mapsto M = C - S$). Sending her ephemeral public key $E$ together with the ciphertext allows Bob to compute $S$ and decrypt with Algorithm 4. Since the secret key here is the bare shared secret $S$, untreated by a KDF, the security of this protocol depends not on the CDHP but rather on the (easier) decisional Diffie–Hellman problem in $\mathcal{G}$.

**Algorithm 3:** Classic ElGamal encryption: Alice encrypts to Bob

**Input:** Bob's public key $B$ and a message $M \in \mathcal{G}$
**Output:** An element $E \in \mathcal{G}$ and a ciphertext $C \in \mathcal{G}$

```
1 function ElGamalEncrypt(B, M)
2     (E, e) ← KeyPair()                          // E = [e]P
3     S ← DH(B, e)                                // B = [b]P ⟹ S = [eb]P
4     C ← M + S                                   // C = M + [eb]P
5     return (E, C)
```

It is important to note that this cryptosystem does not provide semantic security. For example, if $(E_1, C_1)$ and $(E_2, C_2)$ are encryptions of $M_1$ and $M_2$, respectively, then $(E_1 + E_2, C_1 + C_2)$ is a legitimate encryption of $M_1 + M_2$. While this property is desirable for certain applications (such as E-Voting [12]), in most contexts textbook ElGamal cannot be safely used for public-key encryption.

**Algorithm 4:** Classic ElGamal decryption: Bob decrypts from Alice

**Input:** An element $E \in \mathcal{G}$, a ciphertext $M \in \mathcal{G}$, and Bob's private key $b$
**Output:** A plaintext messge $M \in \mathcal{G}$

```
function ElGamalDecrypt((E, C, b))
    S ← DH(E, b)          // E = [e]P ⟹ S = [eb]P
    M ← C − S             // C = M + [eb]P
    return M
```

The homomorphic nature of the scheme is due to the fact that the group law $+$ is being used as the encryption and decryption algorithm. But even if this behaviour is actually desired, requiring the message to be an element of $\mathcal{G}$ poses two further problems. First, it imposes a hard and inconvenient limit on the size of the message space; second, it requires an efficient encoding of messages to group elements (and an efficient decoding to match). At first glance, this second requirement seems straightforward for ElGamal instantiated in $\mathbb{F}_p^\times$, since bitstrings of length $\leq \log_2 p$ can be immediately interpreted as integers in $\mathbb{Z}/p\mathbb{Z}$, and hence elements of $\mathbb{F}_p$; but this embedding does not map into the prime-order subgroups where the protocol typically operates. This complication is worse in the elliptic curve setting, where the message-length limit is even more restrictive.

The modern, semantically secure version of ElGamal encryption is an example of hybrid encryption, best approached through the more general setting of Key Encapsulation Mechanisms (KEMs) and Data Encryption Mechanisms (DEMs) [43,79]. We establish encryption keys using an asymmetric system (the KEM), before switching to symmetric encryption for data transport (the DEM).

Algorithms 5 and 6 illustrate a simple Diffie–Hellman-based KEM. In Algorithm 5, Bob has already generated a long-term keypair $(B, b)$ and published $B$. Alice takes $B$, generates an ephemeral keypair $(E, e)$, completes the Diffie–Hellman on her side, and derives a cryptographic key $K$ from the shared secret $S$. She can use $K$ to encrypt messages to Bob[4], while $E$ encapsulates $K$ for transport. To decapsulate $E$ and decrypt messages from Alice, Bob follows Algorithm 6, completing the Diffie–Hellman on his side and deriving the cryptographic key $K$ from the shared secret $S$.

*Remark 1.* While KEMs provide a convenient API and formalism for key establishment, they cannot always be used as a replacement for plain-old Diffie–Hellman, especially as a component in more complicated protocols.

## 5 Concrete groups and discrete logarithms

So far, everything has been presented in the abstract; but if we want to use any of these schemes in practice, then we need to choose a concrete group $\mathcal{G}$. As we

[4] If Alice immediately encrypts a message under $K$ and sends the ciphertext to Bob with $E$, then this is "hashed ElGamal" encryption (see [1] for a full encryption scheme in this style).

**Algorithm 5:** DH-based KEM: Alice encapsulating to Bob.

**Input:** Bob's public key $B \in \mathcal{G}$
**Output:** A symmetric encryption key $K \in \{0,1\}^n$ and an encapsulation $E \in \mathcal{G}$ of $K$ under $B$

**function** `DHEncapsulate`$(B)$
  $(E, e) \leftarrow$ `KeyPair()` // $E = [e]P$
  $S \leftarrow \mathtt{DH}(B, e)$ // $S = [eb]P$
  $K \leftarrow \mathtt{KDF}(E \parallel S)$ // $K = \mathtt{KDF}(E \parallel [eb]P)$
  **return** $(K, E)$

**Algorithm 6:** DH-based KEM: Bob decapsulating from Alice.

**Input:** An encapsulation $E \in \mathcal{G}$ of a symmetric key $K \in \{0,1\}^n$ under Bob's public key $B \in \mathcal{G}$, and Bob's private key $b \in \mathbb{Z}/N\mathbb{Z}$
**Output:** A symmetric encryption key $K \in \{0,1\}^n$

**function** `DHDecapsulate`$(E, b)$
  $S \leftarrow \mathtt{DH}(E, b)$ // $E = [e]P \implies S = [eb]P$
  $K \leftarrow \mathtt{KDF}(E \parallel S)$ // $E = [e]P \implies K = \mathtt{KDF}([e]P \parallel [eb]P)$
  **return** $K$

noted in §2, the hardness of the DLP (and hence the CDHP) varies according to the representation of $\mathcal{G}$, and may fall far short of the $O(\sqrt{N})$ ideal. Here we give a very brief overview of the main candidate groups for group-based Diffie–Hellman, and DLP-based cryptography in general. We refer the reader to Guillevic and Morain's excellent survey [77] for further detail on discrete logarithm algorithms.

The DLP in prime finite fields, as used by Diffie and Hellman, is subexponential: the General Number Field Sieve [91] solves DLP instances in $\mathbb{F}_p$ in time $L_p[1/3, (64/9)^{1/3}]$.[5] In extension fields of large characteristic, or when the characteristic has a special form, the complexity is lower, while still subexponential (see [77]); in the extreme case of extension fields of tiny characteristic, the DLP is quasipolynomial in the field size [11]. These algorithms can also be used to attack DLPs in algebraic tori, which are compact representations of subgroups of $\mathbb{F}_q^\times$ which offer smaller key sizes and efficient arithmetic [121,92].

Elliptic curves have long been recognised by number theorists as a generalization of the multiplicative group (indeed, both the multiplicative and additive groups can be seen as degenerate elliptic curves; see e.g. [33, §9]). Once Diffie and Hellman had proposed their protocol in a multiplicative group, then, it was perhaps only a matter of time before number theorists proposed elliptic-curve Diffie–Hellman; and within a decade Miller [102] and Koblitz [83] did just this, independently and almost simultaneously. The subexponential finite-field DLP algorithms do not apply to general elliptic curves, and so far we know of no algorithm with complexity better than $O(\sqrt{N})$ for the DLP in a general prime-order elliptic curve. Indeed, the only way we know to make use of the geometric

[5] Recall that $L_X[\alpha, c] = \exp((c + o(1))(\log X)^\alpha (\log\log X)^{1-\alpha})$.

structure for general curves over prime fields is to run generic $O(\sqrt{N})$ algorithms on equivalence classes modulo $\pm 1$, but this only improves the running time by a factor of roughly $\sqrt{2}$ [18].[6] We can do better for some elliptic curves defined over some extension fields [74,139,71], and for some small special classes of curves [100,10,63,127] (notably pairing-friendly curves); but in the more than thirty years since Miller and Koblitz introduced elliptic curve cryptography, this $\sqrt{2}$ speedup represents the only real non-quantum algorithmic improvement for the general elliptic-curve DLP.[7]

Going beyond elliptic curves, a range of other algebraic groups have been proposed for use in cryptography. Koblitz proposed cryptosystems in Jacobians of hyperelliptic curves as an obvious generalization of elliptic curves [84]. Others have since suggested Jacobians of general algebraic curves, and abelian varieties [105,118]; but as the genus of the curve (or the dimension of the abelian variety) grows, index-calculus algorithms become more effective, quickly outperforming generic DLP algorithms. At best, the DLP for curves of fixed genus $\geq 3$ is exponential, but easier than $O(\sqrt{N})$ [75,128,51,73]; at worst, as the genus and field size both tend to infinity, the DLP becomes subexponential [59]. Déchène proposed generalized Jacobians as a bridge between elliptic-curve and finite-field cryptography [47], but these offer no constructive advantage [69].

The groups mentioned above are all *algebraic groups*: elements are represented by tuples of field elements, and group operations are computed using polynomial formulæ. Algebraic groups are well-suited to efficient computation on real-world computer architectures, but they are not the only such groups: another kind consists of class groups of number fields. Buchmann and Williams proposed Diffie–Hellman schemes in class groups of quadratic imaginary orders [31], leading to a series of DLP-based cryptosystems set in more general rings (see [30] for a survey); but ultimately these are all vulnerable to subexponential index-calculus attacks.

In the classical world, therefore, elliptic curves over $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$ and genus-2 Jacobians over $\mathbb{F}_{p^2}$ present the hardest known DLP instances with respect to group size (and hence key size). Elliptic curves over prime fields have become, in a sense, the gold standard to which all other groups are compared.

## 6 Concrete hardness and security levels

It is important to note that our understanding of DLP hardness is not just a matter of theory and sterile asymptotics; all of the algorithms above are backed

[6] More generally, we can work on equivalence classes modulo a (sub)group of automorphisms, as in [55]; but in the case of elliptic curves, for any fixed $\mathbb{F}_q$, there are only two $\overline{\mathbb{F}}_q$-isomorphism classes of curves with automorphisms other than $\pm 1$.

[7] At least, it is the only improvement as far as algorithmic complexity is concerned: implementation and distribution have improved substantially. It is, nevertheless, quite dumbfounding that in over thirty years of cryptographically-motivated research, we have only scraped a tiny constant factor away from the classical algorithmic complexity of the DLP in a generic prime-order elliptic curve over a prime finite field.

by a tradition of experimental work. Recently discrete logarithms have been computed in 768-bit general and 1024-bit special prime fields [82,64], and in 112-bit and 117-bit binary elliptic curve groups [138,17]. A table of various record finite field discrete logarithm computations can be found at [76].

These computations give us confidence in defining cryptographic parameters targeting concrete security levels against classical adversaries. For example, it is generally accepted that DLP-based cryptosystems in $\mathbb{F}_p^\times$ with $\log_2 p \approx 3072$ or in $\mathcal{E}(\mathbb{F}_p)$ with $\log_2 p \approx 256$ for a well-chosen $\mathcal{E}$ should meet a classical approximate 128-bit security level: that is, a classical adversary equipped with current algorithms should spend around $2^{128}$ computational resources to break the system with non-negligable probability.

For quantum adversaries, we know that DLPs can be solved in polynomial time—but we still know relatively little about the concrete difficulty and cost of mounting quantum attacks against DLP-based cryptosystems, let alone against candidate post-quantum systems. For example, we mentioned above that the current record for Shor's factoring algorithm is 21; but to our knowledge, Shor's algorithm for discrete logarithms has never been implemented. Roetteler, Naehrig, Svore and Lauter have estimated the quantum resources required to compute elliptic-curve discrete logarithms [119], which is an important first step.

The situation becomes even murkier for the quantum algorithms we will meet below, including the Kuperberg, Regev, Tani, and Childs–Jao–Soukharev algorithms. We have asymptotic estimates, but no concrete estimates (or real data points, for that matter). It is not clear what the most useful performance metrics are for these algorithms, or how to combine those metrics with classical ones to estimate overall problem difficulty.

For this reason, we will refrain from giving any concrete security estimates or recommendations for key-lengths for the post-quantum systems in the second half of this article. We look forward to detailed theoretical estimates along the lines of [119], and to the eventual development of quantum computers sufficiently large to implement these algorithms and get some actual cryptanalysis done.

## 7 Modern elliptic-curve Diffie–Hellman

At first glance, elliptic-curve cryptography is just finite-field cryptography with a different algebraic group seamlessly swapped in[8], and no theoretical modification. But Miller's original article [102] ends with an interesting observation that departs from the multiplicative group perspective:

> Finally, it should be remarked, that even though we have phrased everything in terms of points on an elliptic curve, that, for the key exchange protocol (and other uses as one-way functions), that only the $x$-coordinate needs to be transmitted. The formulas for multiples of a point cited in the first section make it clear that the $x$-coordinate of a multiple depends only on the $x$-coordinate of the original point.

[8] Not entirely seamlessly: some operations, like hashing into $\mathcal{G}$, become slightly more complicated when we pass from finite fields to elliptic curves (see [109]).

Miller is talking about elliptic curves in Weierstrass models $y^2 = x^3 + ax + b$, where $-(x, y) = (x, -y)$, so $x$-coordinates correspond to group elements modulo sign. The mapping $(m, x(P)) \mapsto x([m]P)$ is mathematically well-defined, because every $[m]$ commutes with $[-1]$; but it can also be computed efficiently.

In Diffie–Hellman, then, instead of using

$$A = [a]P\,, \qquad B = [b]P\,, \qquad S = [ab]P\,,$$

Miller is proposing that we use

$$\begin{aligned} A &= \pm[a]P \\ &= x([a]P)\,, \end{aligned} \qquad \begin{aligned} B &= \pm[b]P \\ &= x([b]P)\,, \end{aligned} \qquad \begin{aligned} S &= \pm[ab]P \\ &= x([ab]P)\,. \end{aligned}$$

Clearly, we lose nothing in terms of security by doing this: the $x$-coordinate CDHP reduces immmediately to the CDHP in the elliptic curve. Given a CDHP oracle for $\mathcal{E}$, we can compute $\pm[ab]P$ from $(\pm P, \pm[a]P, \pm[b]P)$ by choosing arbitrary lifts to signed points on $\mathcal{E}$ and calling the oracle there; conversely, given an $x$-coordinate CDHP oracle, we can solve CDHP instances on $\mathcal{E}$ by projecting to the $x$-line, calling the oracle there, and then guessing the sign on $S$.

The idea of transmitting only the $x$-coordinates may seem advantageous in terms of reducing bandwidth, but in reality, where elliptic curve point keys are systematically compressed to an $x$-coordinate plus a single bit to indicate the "sign" of the corresponding $y$-coordinate, there is little to be gained here beyond avoiding the small effort required for compression and decompression. The real practical value in Miller's idea is that working with only $x$-coordinates is faster, and requires less memory: $x([a]P)$ can be computed from $a$ and $x(P)$ using fewer field operations than would be required to compute $[a]P$ from $a$ and $P$.

This advantage was convincingly demonstrated by Bernstein's `Curve25519` software [14], which put Miller's idea into practice using carefully selected curve parameters optimized for Montgomery's ladder algorithm, which computes the pseudo-scalar multiplications $x(P) \mapsto x([m]P)$ using a particularly efficient and regular differential addition chain [104,41]. The result was not only a clear speed record for Diffie–Hellman at the 128-bit security level, but a new benchmark in design for key exchange software. `Curve25519` is now the Diffie–Hellman to which all others are compared in practice.

The elliptic curve cryptosystems that were standardized in the 1990s and early 2000s, such as the so-called NIST [107] and Brainpool [94] curves, are based on full elliptic curve arithmetic and are not optimized for $x$-only arithmetic. More recently, `Curve25519` and similar systems have been standardized for future internet applications [90]. These systems are also preferred in newer practical applications, such as the Double Ratchet algorithm used for key management within the Signal protocol for end-to-end encrypted messaging [111].

In theory, Miller's idea of working modulo signs (or, more generally, automorphisms) extends to any algebraic group.[9] For example, the quotients of Jacobians

[9] While quotienting by $\pm 1$ is useful in curve-based cryptosystems, it is counterproductive in multiplicative groups of finite fields. There, the pseudo-scalar multiplication

of genus-2 curves by $\pm 1$ are Kummer surfaces. Under suitable parametrizations, these have highly efficient pseudo-scalar multiplications [72], which have been used in high-speed Diffie–Hellman implementations [16,116].

While the $x$-only approach to elliptic curve Diffie–Hellman is particularly useful in practice, it also highlights an important theoretical point: on a formal level, Diffie–Hellman does *not* require a group structure.[10] By this, we mean that the group law never explicitly appears in the protocol—and this is precisely why Diffie–Hellman works on elliptic $x$-coordinates, where there is no group law (but where there *are* maps induced by scalar multiplication).

The group structure is still lurking behind the scenes, of course. It plays several important roles:

1. *Correctness.* The group law gives an easy proof that the pseudo-scalar multiplication operations $(m, x(P)) \mapsto x([m]P)$ exist and commute.
2. *Efficiency.* The group law induces biquadratic relations on $x$-coordinates that we use to efficiently compute pseudo-scalar multiplications using suitable differential addition chains [15].
3. *Security.* The hardness of the CDHP in the full group underwrites the hardness of the $x$-coordinate CDHP.

*Remark 2.* Can we do without a group entirely? Heading into the pure abstract, we can consider a Diffie–Hellman protocol with minimal algebraic structure. For example, we could take a set $\mathcal{X}$ of public keys in place of the group $\mathcal{G}$, and sample private keys from a set $\mathcal{F}$ of functions $\mathcal{X} \to \mathcal{X}$ defined by the property[11]

$$(a \circ b)(P) = (b \circ a)(P) \quad \text{for all} \quad a, b \in \mathcal{F}\,.$$

The associated Diffie–Hellman protocol is then defined by

$$A = a(P)\,, \qquad B = b(P)\,, \qquad S = a(b(P)) = b(a(P))\,.$$

We need $\mathcal{F}$ to be large enough to prevent brute force attacks on $S$; we must be able to efficiently sample functions from $\mathcal{F}$, and evaluate them at elements of $\mathcal{X}$; and we need to justify the hardness of the associated CDHP. An algebraic structure on $\mathcal{F}$ may not be strictly necessary to ensure all of this, but it certainly makes life easier.

is $(m, P + 1/P) \mapsto P^m + 1/P^m$; computing this is slightly slower than computing simple exponentiations, and saves no space at any point.

[10] Buchmann, Scheidler, and Williams later proposed what they claimed was the first group-less key exchange in the infrastructure of real quadratic fields [29]. Mireles Morales investigated the infrastructure in the analogous even-degree hyperelliptic function field case [103], relating it to a subset of the class group of the field; in view of his work, it is more appropriate to describe infrastructure key exchange as group-based. In any case, coming nearly a decade after Miller, this would not have been the first non-group Diffie–Hellman.

[11] If we require this property to hold for *all* $P$ in $\mathcal{X}$, then $\mathcal{F}$ is a *commutative magma*. Diffie–Hellman protocols where $\mathcal{F}$ is equipped with a semigroup or semiring structure have been investigated [99], though the results are only of theoretical interest.

## 8 Principal homogeneous spaces

At the time of writing, the closest thing we have to a post-quantum analogue of Diffie–Hellman comes from *isogeny-based cryptography*, whose origins go back to Couveignes' "Hard Homogeneous Spaces" manuscript [42]. This went unpublished for ten years, before appearing online more or less at the same time as its ideas were independently rediscovered by Rostovtsev and Stolbunov [120].

Couveignes' work is a convenient framework for reasoning about isogeny-based cryptosystems: the hard detail on class groups and isogeny classes is abstracted away into groups acting on sets. We warn the reader that from now on we will mostly be working with abelian groups written *multiplicatively*, which we denote by $\mathfrak{G}$ in accordance with the convention from §1.

Recall that a (left) *action* of a group $\mathfrak{G}$ on a set $\mathcal{X}$ is a mapping $\mathfrak{G} \times \mathcal{X} \to \mathcal{X}$, written $(\mathfrak{g}, P) \mapsto \mathfrak{g} \cdot P$, compatible with the group operation in $\mathfrak{G}$: that is,

$$\mathfrak{g}_1 \cdot (\mathfrak{g}_2 \cdot P) = (\mathfrak{g}_1 \mathfrak{g}_2) \cdot P \quad \text{for all } \mathfrak{g}_1, \mathfrak{g}_2 \in \mathfrak{G} \text{ and } P \in \mathcal{X}\,.$$

In our case $\mathfrak{G}$ is abelian, so $\mathfrak{g}_1 \cdot (\mathfrak{g}_2 \cdot P) = \mathfrak{g}_2 \cdot (\mathfrak{g}_1 \cdot P)$ for all $\mathfrak{g}_1$, $\mathfrak{g}_2$, and $P$.

**Definition 3 (PHS).** *A* principal homogeneous space (PHS) *for an abelian group $\mathfrak{G}$ is a set $\mathcal{X}$ equipped with a simple, transitive action of $\mathfrak{G}$: that is, for any $P$ and $Q$ in $\mathcal{X}$, there is a* unique *$\mathfrak{g}$ in $\mathfrak{G}$ such that $Q = \mathfrak{g} \cdot P$. Equivalently, for every $P$ in $\mathcal{X}$, the map $\varphi_P : \mathfrak{G} \to \mathcal{X}$ defined by $\mathfrak{g} \mapsto \mathfrak{g} \cdot P$ is a bijection.*

*Example 1.* The trivial example of a PHS is a group acting on itself via its own group operation: that is, $\mathcal{X} = \mathfrak{G}$, with $\mathfrak{g} \cdot \mathfrak{a} = \mathfrak{g}\mathfrak{a}$.

*Example 2.* The classic first example of a nontrivial PHS is a vector space acting by translation on its underlying affine space.

Example 2 illustrates a classic informal definition: a PHS is a group whose identity element has been forgotten (mislaid, not omitted). Affine spaces have no distinguished "origin"; on the other hand, as soon as one is (arbitrarily) chosen, then each point defines an implicit displacement vector, and we can identify the affine space with a vector space. More generally, for each $P$ in $\mathcal{X}$, if we define $\varphi_P(\mathfrak{g}_1)\varphi_P(\mathfrak{g}_2) := \varphi_P(\mathfrak{g}_1\mathfrak{g}_2)$ then we get a well-defined group structure on $\mathcal{X}$; in fact, we get a different group structure for each choice of $P$. The idea therefore is not so much that the identity element has been forgotten, yet might still be remembered; it is rather that every single element is an equally plausible identity.

*Example 3.* Let $\mathcal{X}$ be the set of points on a curve $\mathcal{C}$ of genus 1, and let $\mathfrak{G} = \mathrm{Pic}^0(\mathcal{C})$ be the group of degree-0 divisor classes on $\mathcal{C}$. By the Riemann–Roch theorem, for every class $[D]$ in $\mathrm{Pic}^0(\mathcal{C})$ and point $P$ on $\mathcal{C}$, there exists a unique $P_D$ on $\mathcal{C}$ such that $[D] = [P_D - P]$. We therefore have an explicit action of $\mathfrak{G}$ on $\mathcal{X}$, defined by $[D] \cdot P = P_D$. If we fix a choice of distinguished "base point" $O$ in $\mathcal{X}$, then we can identify each class $[D]$ with the point $[D] \cdot O$, and thus, by transport of structure, we get a group law on $\mathcal{X}$. (Cognoscenti will recognise the definition of the group law on an arbitrary elliptic curve via the Picard group.)

Our final example of a PHS is fundamental in isogeny-based cryptography. It is far more complicated to define than Examples 1, 2, and 3; we will give an extremely brief description here, returning to it in greater detail in §13 and §14.

*Example 4.* Let $q$ be a prime power and $t$ an integer with $|t| \leq 2\sqrt{q}$; let $\mathcal{O}_K$ be the ring of integers of the imaginary quadratic field $K = \mathbb{Q}(\sqrt{\Delta})$, where $\Delta := t^2 - 4q$. Let $\mathcal{X}$ be the set of $\mathbb{F}_q$-isomorphism classes of elliptic curves $\mathcal{E}/\mathbb{F}_q$ whose $\mathbb{F}_q$-endomorphism ring is isomorphic to $\mathcal{O}_K$ (and where the image of the Frobenius endomorphism of $\mathcal{E}$ in $\mathcal{O}_K$ has trace $t$). Then $\mathcal{X}$ is a PHS under the ideal class group $\mathfrak{G} = \mathrm{Cl}(\mathcal{O}_K)$ of $\mathcal{O}_K$, with ideals acting by $\mathfrak{a} \cdot \mathcal{E} = \mathcal{E}/\mathcal{E}[\mathfrak{a}]$, where $\mathcal{E}[\mathfrak{a}]$ is the intersection of the kernels of the endomorphisms in $\mathfrak{a}$. This PHS is central to the theory of Complex Multiplication; there is also a well-developed algorithmic theory for it, used to compute fundamental number-theoretic objects such as modular and Hilbert class polynomials (see e.g. [27]).

Example 4 highlights another view of PHSes: we can consider $\mathcal{X}$ as a version of $\mathfrak{G}$ whose structure is hidden by the maps $\varphi_P$. In this case, the elements of $\mathcal{X}$ are $j$-invariants, and (when the class group is sufficiently large) look like random elements of $\mathbb{F}_q$; the class group itself has no such encoding.

## 9 Hard homogeneous spaces

Let $\mathcal{X}$ be a PHS under $\mathfrak{G}$. From now on we assume we can efficiently compute group operations, evaluate actions, test equality, and hash elements of $\mathfrak{G}$ and $\mathcal{X}$. We also assume we can uniformly randomly sample elements of $\mathfrak{G}$. Figure 1 illustrates the two interesting computational problems in this setting, which Couveignes called *vectorization* (Definition 4) and *parallelization* (Definition 5).

**Definition 4 (Vectorization).** *The* vectorization *problem in a PHS $\mathcal{X}$ under $\mathfrak{G}$ is, given $P$ and $Q$ in $\mathcal{X}$, to compute the unique $\mathfrak{g}$ in $\mathfrak{G}$ such that $Q = \mathfrak{g} \cdot P$.*

**Definition 5 (Parellelization).** *The* parallelization *problem in a PHS $\mathcal{X}$ under $\mathfrak{G}$ is, given $P$, $A$, and $B$ in $\mathcal{X}$, to compute the unique $S$ in $\mathcal{X}$ such that $S = (\mathfrak{a}\mathfrak{b}) \cdot P$ where $A = \mathfrak{a} \cdot P$ and $B = \mathfrak{b} \cdot P$. (Note that then $S = \mathfrak{a} \cdot B = \mathfrak{b} \cdot A$.)*

**Definition 6 (HHS).** *Let $\mathcal{X}$ be a PHS under $\mathfrak{G}$. We say $\mathcal{X}$ is a* hard homogeneous space (HHS) *if the action of $\mathfrak{G}$ on $\mathcal{X}$ is efficiently computable, but the vectorization and parallelization problems are computationally infeasible.*

The names "vectorization" and "parallelization" are intuitive in the context of Example 2: vectorization is computing the displacement vector between points $P$ and $Q$ in the space $\mathcal{X}$, while parallelization is completing the parallelogram with vertices $P$, $A$, and $B$. These are routine operations in vector and affine spaces, so the PHS of Example 2 is typically not something that we would consider an HHS. Similarly, the PHS of Example 3 is not an HHS, because we can always vectorize by formally subtracting points to form a degree-0 divisor. Couveignes suggested that the PHS of Example 4 might be an HHS—and with the current state of classical and quantum class group and isogeny algorithms, it is.

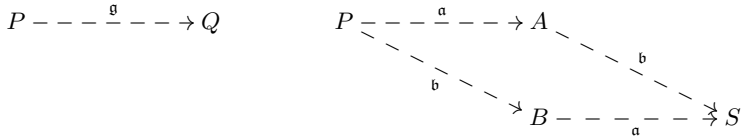


**Fig. 1.** Vectorization (left: finding the unique $\mathfrak{g}$ such that $Q = \mathfrak{g} \cdot P$) and parallelization (right: computing the unique $S$ such that $S = \mathfrak{b} \cdot A = \mathfrak{a} \cdot B = (\mathfrak{ab}) \cdot P$). The dashed arrows denote actions of the unknown group elements $\mathfrak{g}$, $\mathfrak{a}$, and $\mathfrak{b}$.

## 10 Cryptography in hard homogeneous spaces

On a purely symbolic level, the vectorization problem $(P, \mathfrak{g} \cdot P) \mapsto \mathfrak{g}$ in a PHS bears an obvious formal resemblance to the DLP $(P, [x]P) \mapsto x$ in a group, just as the parallelization problem $(P, \mathfrak{a} \cdot P, \mathfrak{b} \cdot P) \mapsto \mathfrak{ab} \cdot P$ resembles the CDHP $(P, [a]P, [b]P) \mapsto [ab]P$. The presence of abelian groups in each problem suggests deeper connections—connections that do not necessarily exist. Indeed, we saw above that parallelization in a PHS is an implicit computation of the group law, while the Diffie–Hellman operation in a group is something completely different.

But irrespective of the relationship between parallelizations and CDHPs, this syntactical resemblance allows us to define a cryptosystem analogous to Diffie–Hellman. Algorithms 7 and 8 define Couveignes' key exchange in the HHS setting, with security depending on the hardness of parallelization.

**Algorithm 7:** Key generation for cryptosystems in an HHS $\mathcal{X}$ under $\mathfrak{G}$, with a fixed base point $P$ in $\mathcal{X}$

**Input:** ()
**Output:** A private-public keypair $(Q, \mathfrak{g}) \in \mathfrak{G} \times \mathcal{X}$ s.t. $Q = \mathfrak{g} \cdot P$

```
function KeyPair()
    g ← Random(G)          // g is sampled uniformly at random from G
    Q ← g · P
    return (Q, g)
```

**Algorithm 8:** Diffie–Hellman in an HHS $\mathcal{X}$ under a group $\mathfrak{G}$

**Input:** A private key $\mathfrak{g}_A \in \mathfrak{G}$ and a public key $Q_B \in \mathcal{X}$, each generated by calls to Algorithm 7
**Output:** A shared secret value $S \in \mathcal{X}$

```
function DH(Q_B, g_A)
    S ← g_A · Q_B
    return S
```

Algorithms 7 and 8 immediately raise some important restrictions on the kinds of $\mathfrak{G}$ and $\mathcal{X}$ that we can use. The first is that we need some kind of canonical representation for elements of $\mathcal{X}$, to ensure that Alice and Bob can derive equal shared cryptographic keys from the shared secret $S$. This property is also important in settings where public keys are required to be unique for a given private key. We also need to be able to efficiently draw uniformly random samples from $\mathfrak{G}$; and then, given a random element of $\mathfrak{G}$, we need to be able to efficiently compute its action on arbitrary elements of $\mathcal{X}$. An alternative approach is to repeatedly randomly sample from a subset of efficiently-computable elements of $\mathfrak{G}$, building a sequence of such elements to be used as the secret key, with the action of the key being the composition of the action of its components. This approach requires an argument that the distribution of these compositions is sufficiently close to the uniform distribution on the whole of $\mathfrak{G}$. Both approaches are relevant in isogeny-based cryptography, as we will see in §14.

Many CDHP-based cryptosystems have obvious HHS analogues. We can define an HHS-based KEM (and implicitly, a hashed-ElGamal-type public key encryption scheme) along the lines of Algorithms 5 and 6, by replacing the calls to Algorithms 1 and 2 with calls to Algorithms 7 and 8, respectively. But not all DLP-based protocols have HHS-based analogues: for example, the obvious HHS analogue of Schnorr's signature scheme [122] would appear to require an efficient (decisional) parallelization algorithm in order to verify signatures.

HHS-Diffie–Hellman is *not* a natural generalization of group-Diffie–Hellman. As we noted in §7, in group-DH, we have a ring (integers modulo $N$) acting on the group $\mathcal{G}$; the composition operation at the heart of DH is ring multiplication, but the ring only forms a group under addition. Formally, in group-DH we only use the fact that the scalars form a commutative magma; but algorithmically, we exploit the fact that the elements form an abelian group and the scalars form a commutative ring, mapping addition in the ring onto the group law, in order to efficiently evaluate scalar multiplications using addition chains.

More concretely, we noted in §8 that the maps $\varphi_P : \mathfrak{g} \mapsto \mathfrak{g} \cdot P$ can be seen as hiding the group $\mathfrak{G}$ in $\mathcal{X}$. The parallelization $(P, \mathfrak{a} \cdot P, \mathfrak{b} \cdot P) \mapsto \mathfrak{ab} \cdot P$ can be written as $(\varphi_P(1), \varphi_P(\mathfrak{a}), \varphi_P(\mathfrak{b})) \mapsto \varphi_P(\mathfrak{ab})$; that is, parallelization computes the group law in the hidden representation of $\mathfrak{G}$ in $\mathcal{X}$ corresponding to $P$. From this perspective, HHS-Diffie–Hellman is a hidden version of the ridiculous key exchange where the shared secret is the product of the two public keys: obviously, without the hiding, this offers no security whatsoever.

## 11 Vectorization and parallelization

To argue about the security of the schemes in §10, we must address the following questions: how hard are vectorization and parallelization? What is the relationship between these problems, and to what extent does our common intuition relating the DLP and CDHP carry over to vectorization and parallelization in the PHS setting?

It might seem excessive to require the simple and transitive action of a PHS in all this: we could relax and set up the same cryptosystems with a group $\mathfrak{G}$ acting on a set $\mathcal{X}$ in any old way. While we might lose uniqueness and/or existence of vectorizations and parallelizations, many of the arguments in this section would still go through. However, using PHSes instead of general group actions makes one thing particularly simple: the proof of random self-reducibility for vectorization and parallelization is identical to the usual arguments for groups, which we sketched in §5. More precisely: if an algorithm successfully solves vectorizations in $(\mathfrak{G}, \mathcal{X})$ with a probability of $1/M$, then we can solve *any* vectorization in $(\mathfrak{G}, \mathcal{X})$ with an expected $M$ calls to the algorithm. Given a target vectorization $(P, Q = \mathfrak{g} \cdot P) \mapsto \mathfrak{g}$, we attempt to solve $(\mathfrak{a} \cdot P, \mathfrak{b} \cdot Q) \mapsto \mathfrak{abg}$ for randomly chosen $\mathfrak{a}$ and $\mathfrak{b}$ in $\mathfrak{G}$; we expect to land in the subset of inputs where the algorithm succeeds within $M$ attempts, and then recovering $\mathfrak{g}$ from $(\mathfrak{a}, \mathfrak{b}, \mathfrak{abg})$ is trivial. This means that the average- and worst-case difficulties for vectorization are equivalent; a similar argument yields the same result for parallelization.

Now, consider the relationship between vectorization and parallelization. If we can solve vectorizations $(P, \mathfrak{g} \cdot P) \mapsto \mathfrak{g}$, then we can can solve parallelizations $(P, \mathfrak{a} \cdot P, \mathfrak{b} \cdot P) \mapsto \mathfrak{ab} \cdot P$, so parallelization is notionally easier than vectorization.

As we have seen, the parallelization operation $(P, \mathfrak{a} \cdot P, \mathfrak{b} \cdot P) \mapsto \mathfrak{ab} \cdot P$ acts as the group law induced on $\mathcal{X}$ by $\mathfrak{G}$ when elements are hidden by $\varphi_P : \mathfrak{g} \mapsto \mathfrak{g} \cdot P$. Given a parallelization oracle for a PHS $(\mathfrak{G}, \mathcal{X})$ with respect to one fixed base point $P$ (call this *$P$-parallelization*), we can view $\mathcal{X}$ as an efficiently computable group, and thus apply any black-box group algorithm to $\mathcal{X}$. Further, given an efficient $P$-parallelization algorithm, the map $\varphi_P$ becomes an efficiently computable group homomorphism. Even further, if we have a $P$-parallelization oracle for *all* $P$ in $\mathcal{X}$, then the mapping $(\mathfrak{g}, P) \mapsto \mathfrak{g} \cdot P$ becomes an efficiently computable bilinear pairing $\mathfrak{G} \times \mathcal{X} \to \mathcal{X}$ (viewing $\mathcal{X}$ as a version of $\mathfrak{G}$ hidden by one $\varphi_O$).

The efficient homomorphism $\varphi_P : \mathfrak{G} \to \mathcal{X}$ implied by a $P$-parallelization oracle is of course an isomorphism (because $\#\mathcal{X} = \#\mathfrak{G}$), but its inverse is not necessarily efficient[12]: if it were, then we could solve vectorizations $(P, \mathfrak{g} \cdot P) \mapsto \mathfrak{g}$ because $\mathfrak{g} = \varphi_P^{-1}(\mathfrak{g} \cdot P)$. Conversely, if we can vectorize with respect to $P$, then we can invert $\varphi_P$: the preimage $\varphi_P^{-1}(T)$ of any $T$ in $\mathcal{X}$ is the vectorization of $T$ with respect to $\varphi_P(1)$. Parallelization therefore yields an efficient isomorphism in one direction, while vectorization yields the inverse as well.

In the case where a group $\mathcal{G}$ has prime order, we can use a CDHP oracle to view $\mathcal{G}$ as a *black-box field* in the sense of Boneh and Lipton [20]. Given a base point $P$ in $\mathcal{G}$, each element $[a]P$ of $\mathcal{G}$ is implicitly identified with its discrete logarithm $a$. The Diffie–Hellman operation $(P, [a]P, [b]P) \mapsto [ab]P$ becomes an implicit multiplication, allowing us to view $\mathcal{G}$ as a model of $\mathbb{F}_N$, and thus to apply various subexponential and polynomial-time reductions from the DLP to the CDHP in $\mathcal{G}$ (as we noted in §3). A parallelization oracle for $(\mathfrak{G}, \mathcal{X})$, however, only allows us to view $\mathcal{X}$ as a black-box group, not a black-box field; we therefore have no equivalent of the den Boer or Maurer reductions in the HHS setting.

[12] The term *one-way group action* is used for the HHS framework in [34] and [22]. This hints at a more general setting, where actions are not necessarily simple or transitive.

The separation between vectorization and parallelization therefore seems more substantial than the somewhat thin and rubbery separation between the DLP and CDHP. However, we would still like to have some upper bounds for the hardness of these problems. For vectorization, we can give some algorithms.

In the classical setting, Couveignes noted that Shanks' baby-step giant-step (BSGS) and Pollard's probabilistic algorithms for DLPs in groups extend to vectorizations in PHSes. Algorithm 9 is a BSGS analogue for a PHS $\mathcal{X}$ under $\mathfrak{G}$.[13] Given $P$ and $Q = \mathfrak{g} \cdot P$ in $\mathcal{X}$ and a generator $\mathfrak{e}$ for (a subgroup of) $\mathfrak{G}$, Algorithm 9 computes an exponent $x$ such that $\mathfrak{g} = \mathfrak{e}^x$, if it exists (if $\mathfrak{g}$ lies outside the subgroup generated by $\mathfrak{e}$, then the algorithm will fail and return $\perp$).

**Algorithm 9:** BSGS for a PHS $\mathcal{X}$ under (a subgroup of) a group $\mathfrak{G}$.

**Input:** Elements $P$ and $Q$ in $\mathcal{X}$, and an element $\mathfrak{e}$ in $\mathfrak{G}$
**Output:** $x$ such that $Q = \mathfrak{e}^x \cdot P$, or $\perp$

```
function BSGS(P, Q, 𝔢)
    β ← ⌈√(#⟨𝔢⟩)⌉        // May be replaced with an estimate if #⟨𝔢⟩ not known
    S ← {}                  // Hash table: keys in 𝒳, values in ℤ/Nℤ
    T ← P
    for i in [0, ..., β] do
        S[T] ← i
        T ← 𝔢 · T
    (T, 𝔠) ← (Q, 𝔢^β)
    for j in [0, ..., β] do
        if T ∈ S then
            i ← S[T]
            return i − jβ                                   // 𝔢^{jβ} · Q = 𝔢^i · P
        T ← 𝔠 · T
    return ⊥
```

Vectorization in a PHS $\mathcal{X}$ under $\mathfrak{G}$ can always be solved classically in time and space $\widetilde{O}(\sqrt{\#\mathfrak{G}})$ using Algorithm 9 and random self-reducibility, provided a generator of a polynomial-index subgroup of $\mathfrak{G}$ is known. Algorithm 9 does more than what is required: it returns not just the desired vectorization $\mathfrak{g}$, but the discrete logarithm of $\mathfrak{g}$ with respect to $\mathfrak{e}$. This betrays the fact that Algorithm 9 is just a black-box group algorithm operating on a hidden version of $\mathfrak{G}$.

Pollard's algorithms also generalize easily to the HHS setting, because we can compute the pseudorandom walks using only translations, or "shifts", by group elements. These translations in the group setting can be replaced by actions by

[13] Algorithm 9 becomes the usual BSGS for DLPs in $\mathfrak{G} = \langle \mathfrak{e} \rangle$ if we let $\mathcal{X} = \mathfrak{G}$ (with the group operation as the action), let $P = 1$, and let $Q$ be the discrete log target.

the same elements in the HHS setting. The space complexity of vectorization can thus be reduced to as little as $O(1)$ for the same time complexity.

Moving to the quantum setting: despite its resemblance to the DLP, vectorization cannot be solved with Shor's algorithm. In fact, vectorization is an instance of the *abelian hidden shift problem* [135]: given functions $f$ and $g$ such that $f(x \cdot s) = g(x)$ for all $x$ and some "shift" $s$, find $s$. The hidden shift instance corresponding to the vectorization instance $(P, Q = \mathfrak{g} \cdot P)$ has $f = \varphi_P : \mathfrak{G} \to \mathcal{X}$, $g = \varphi_Q : \mathfrak{G} \to \mathcal{X}$, and $s = \mathfrak{g}$. Kuperberg reduces the abelian hidden shift problem to an instance of the dihedral hidden subgroup problem, which is then solved with a quantum algorithm with a query complexity of $L_N[1/2, c]$, where $c = \sqrt{2}$ according to [36]. Kuperberg's original algorithm [88] uses subexponential space; Regev's simpler algorithm [115] uses polynomial quantum space; and Kuperberg's most recent work [89] uses linear quantum space, but subexponential classical space. More detailed perspectives on these algorithms in the context of the isogeny class HHS appear in [34,46,22].

## 12 The difficulty of exploiting subgroup structures

Moving back to the abstract: when we think about DLPs, in black-box or in concrete groups, we implicitly and systematically apply the Pohlig–Hellman–Silver algorithm to reduce to the prime-order case. It is interesting to note that for PHSes, no such reduction is known: it appears difficult to exploit the subgroup structure of $\mathfrak{G}$ when solving vectorization problems in $\mathcal{X}$.[14]

Algorithm 10 presents the Pohlig–Hellman–Silver algorithm for discrete logarithms in a group $\mathfrak{G}$ whose order has known factorization $N = \prod_i N_i^{e_i}$. Line 9 applies a DLP-solving algorithm (like BSGS, Pollard $\rho$, or a specialized algorithm for a concrete group) in the order-$N_i$ subgroup of $\mathfrak{G}$. If the factorization is complete and the $N_i$ are prime, then the global DLP is reduced to a polynomial number smaller prime-order sub-DLPs.

The key steps involve producing the subgroup DLPs. Lines 3 and 4 project the DLP instance $(\mathfrak{g}, \mathfrak{h})$ into the order-$N_i^{e_i}$ subgroup of $\langle \mathfrak{g} \rangle$. Lines 7 and 8 then produce a DLP instance in the order-$N_i$ subgroup. This is always done by exponentiation by $N/N_i^{e_i}$ and $N_i$; indeed, this is the only way that the factors $N_i$ are used in the algorithm.

In the PHS setting, subgroup DLPs should be replaced with subgroup vectorizations. Line 9 could be replaced with a call to Algorithm 9, using $\mathfrak{e}^{N_i^{e_i-1}}$ as the subgroup generator; the problem is to produce a vectorization instance in a sub-PHS acted on by the corresponding subgroup. We cannot naively concatenate "$\cdot P$" (or "$\cdot Q$") to most lines of the algorithm to turn group elements and operations into PHS elements and operations: Line 4, for example, would

[14] It might seem odd that some black-box group algorithms like BSGS and Pollard $\rho$ adapt easily to PHSes, but not others like Pohlig–Hellman. But looking closer, BSGS and Pollard $\rho$ in groups only require translations, and not a full group law. We can therefore see BSGS and Pollard $\rho$ not as black-box group algorithms, but rather as black-box PHS algorithms that are traditionally applied with $\mathcal{X} = \mathfrak{G}$.

**Algorithm 10:** Pohlig–Hellman–Silver for a group $\mathfrak{G}$ whose order has known (partial) factorization.

**Input:** An element $\mathfrak{e}$ of $\mathfrak{G}$, a $\mathfrak{g}$ in $\langle\mathfrak{e}\rangle$, and $((N_1, e_1), \ldots, (N_n, e_n))$ such that $\#\mathfrak{G} = N = \prod_{i=1}^{n} N_i^{e_i}$, with the $N_i$ pairwise coprime and the $e_i > 0$.
**Output:** $x$ such that $\mathfrak{g} = \mathfrak{e}^x$

**function** `PohligHellman`$(\mathfrak{e}, \mathfrak{g}, ((N_1, e_1), \ldots, (N_n, e_n)))$
  **for** $1 \le i \le n$ **do**
    $\mathfrak{e}_i \leftarrow \mathfrak{e}^{N/N_i^{e_i}}$
    $\mathfrak{g}_i \leftarrow \mathfrak{g}^{N/N_i^{e_i}}$
    $x_i \leftarrow 0$
    **for** $j$ *in* $(e_i - 1, \ldots, 0)$ **do**
      $\mathfrak{s} \leftarrow \mathfrak{e}_i^{(N_i)^j}$ `// s is in the order-`$N_i$ `subgroup`
      $\mathfrak{t} \leftarrow \mathfrak{s}^{-x_i} \cdot \mathfrak{g}_i^{(N_i)^j}$ `// t is in the order-`$N_i$ `subgroup`
      $y \leftarrow \log_{\mathfrak{s}}(\mathfrak{t})$ `// Use e.g. baby-step giant-step`
      $x_i \leftarrow x_i N_i + y$
  $x \leftarrow$ `CRT`$(\{(x_i, N_i^{e_i}) : 1 \le i \le n\})$
  **return** $x$

require computing $\mathfrak{g}^{N/N_i^{e_i}} \cdot P$ from $Q = \mathfrak{g} \cdot P$, $P$, and $\mathfrak{g}$, but this amounts to an iterated parallelization—and parallelization is supposed to be hard in an HHS.

A thorough investigation of the possibility and difficulty of exploiting subgroup structures for vectorization and parallelization would require working with subgroup actions on quotient spaces; we do not have room to discuss this here. We note, however, that in some protocols a limited number of exploitable parallelizations are provided by the protocol itself, as in the group-based protocols subject to Cheon's attack [35], and this should have some consequences for the security of any HHS analogues of these protocols.

## 13 A quick introduction to isogenies

This section provides enough background on isogenies and endomorphisms of elliptic curves to make sense of the HHS from Example 4 before we describe cryptosystems based on it in §14. We also want to fill in some background on supersingular curves before we need it in §15. We assume a basic familiarity with the arithmetic of elliptic curves; readers familiar with isogenies and isogeny graphs can safely skip this section. As a mathematical reference, we suggest [126] and [86]; for greater detail focused on the cryptographic use case, see [44].

We want to talk about relationships between elliptic curves over a fixed finite field $\mathbb{F}_q$, where $q$ is a power of some prime $p$. We can assume that $p \neq 2$ or 3, to simplify, though the theory applies to those cases as well. We will work with elliptic curves as short Weierstrass models $\mathcal{E} : y^2 = x^3 + ax + b$, with $a$ and $b$ in $\mathbb{F}_q$: in practice we might compute using other curve models (many isogeny-based cryptosystem implementations have preferred Montgomery arithmetic [41]), but

since we end up working with curves up to $\mathbb{F}_q$-isomorphism, and every curve is $\mathbb{F}_q$-isomorphic to a short Weierstrass model, we lose nothing in restricting to this simple and universal curve shape in this article. The $m$-torsion $\mathcal{E}[m]$ of $\mathcal{E}$ is the subgroup of points $P$ such that $[m]P = 0_\mathcal{E}$.

A *homomorphism* $\phi : \mathcal{E} \to \mathcal{E}'$ is, by definition[15], a rational map such that $\phi(0_\mathcal{E}) = 0_{\mathcal{E}'}$. Homomorphisms induce homomorphisms on groups of points [126, §III.4], but not every homomorphism of groups of points is induced by a homomorphism of curves. An $\mathbb{F}_q$-homomorphism is one that is defined over $\mathbb{F}_q$: that is, the rational functions defining it as a rational map have coefficients in $\mathbb{F}_q$. Every homomorphism here will be defined over $\mathbb{F}_q$, unless explicitly stated otherwise.

*Isogenies* are nonzero homomorphisms of elliptic curves. If there is an isogeny from $\mathcal{E}$ to $\mathcal{E}'$, then we say that $\mathcal{E}$ and $\mathcal{E}'$ are *isogenous*. We will see below that for each isogeny $\mathcal{E} \to \mathcal{E}'$ there is a dual isogeny $\mathcal{E}' \to \mathcal{E}$, so isogeny is an equivalence relation on elliptic curves.

*Isomorphisms* are invertible homomorphisms. The $j$-invariant of a curve $\mathcal{E} : y^2 = x^3 + ax + b$ is $j(\mathcal{E}) = 1728 \cdot \frac{4a^3}{4a^3+27b^2}$; two curves $\mathcal{E}$ and $\mathcal{E}'$ are $\overline{\mathbb{F}}_q$-isomorphic if and only if $j(\mathcal{E}) = j(\mathcal{E}')$. We need to work with the stronger notion of $\mathbb{F}_q$-isomorphism, where the $j$-invariant does not tell the whole story. Curves that are $\overline{\mathbb{F}}_q$-isomorphic but not $\mathbb{F}_q$-isomorphic are called *twists*. The most important example is the *quadratic twist*, which is isomorphic over $\mathbb{F}_{q^2}$ but not $\mathbb{F}_q$: the quadratic twist of $\mathcal{E} : y^2 = x^3 + ax + b$ is $\mathcal{E}' : v^2 = u^3 + \mu^2 au + \mu^3 b$, where $\mu$ is any nonsquare in $\mathbb{F}_q$ (the choice of nonsquare makes no difference up to $\mathbb{F}_q$-isomorphism, which is why we say *the* rather than *a* quadratic twist). The isomorphism $\tau : \mathcal{E} \to \mathcal{E}'$ is defined by $(x, y) \mapsto (u, v) = (\mu x, \mu^{3/2} y)$; this is clearly not defined over $\mathbb{F}_q$, yet $j(\mathcal{E}) = j(\mathcal{E}')$. The quadratic twist of a curve $\mathcal{E}$ is its only twist, up to $\mathbb{F}_q$-isomorphism, unless $j(\mathcal{E}) = 0$ or 1728 (in which case there may be four or two more twists, respectively). Specifying an $\mathbb{F}_q$-isomorphism class therefore comes down to specifying a $j$-invariant and a choice of twist.

*Endomorphisms* are homomorphisms from a curve to itself. The endomorphisms of a given curve $\mathcal{E}$ form a ring $\mathrm{End}(\mathcal{E})$, with the group law on $\mathcal{E}$ inducing addition of endomorphisms and composition of endomorphisms corresponding with multiplication. The structure of the set of isogenies from $\mathcal{E}$ to other curves is deeply connected to the structure of $\mathrm{End}(\mathcal{E})$, and vice versa.

The scalar multiplication maps $[m]$ are endomorphisms, so $\mathrm{End}(\mathcal{E})$ always contains a copy of $\mathbb{Z}$. It also includes the *Frobenius* endomorphism $\pi : (x, y) \mapsto (x^q, y^q)$, which satisfies the quadratic equation $\chi(X) := X^2 - tX + q = 0$ for some integer $t$ in the *Hasse interval* $[-2\sqrt{q}, 2\sqrt{q}]$; we call $t$ the *trace* of Frobenius (and of $\mathcal{E}$). Since points in $\mathcal{E}(\mathbb{F}_q)$ are precisely the points fixed by $\pi$, we have $\#\mathcal{E}(\mathbb{F}_q) = \chi(1) = q + 1 - t$. If $\mathcal{E}'$ is the quadratic twist of $\mathcal{E}$ and we pull back the

[15] An elliptic curve is by definition a pair $(\mathcal{E}, 0_\mathcal{E})$, where $\mathcal{E}$ is a curve of genus 1 and $0_\mathcal{E}$ is a distinguished point on $\mathcal{E}$ (which becomes the identity element of the group of points; cf. Example 3); so it makes sense that a morphism $(\mathcal{E}, 0_\mathcal{E}) \to (\mathcal{E}', 0_{\mathcal{E}'})$ in the category of elliptic curves should be a mapping of algebraic curves $\mathcal{E} \to \mathcal{E}'$ preserving the distinguished points, that is, mapping $0_\mathcal{E}$ onto $0_{\mathcal{E}'}$.

Frobenius on $\mathcal{E}'$ to an endomorphism on $\mathcal{E}$ via the twisting isomorphism, then the result is $-\pi$, so the trace of $\mathcal{E}'$ is the negative of the trace of $\mathcal{E}$.

Now consider the set of all elliptic curves over $\mathbb{F}_q$. Tate's theorem tells us that two elliptic curves are $\mathbb{F}_q$-isogenous if and only if they have the same trace (and hence the same number of rational points). This means that the set of all elliptic curves is partitioned into $\mathbb{F}_q$-isogeny classes, indexed by the integers in the Hasse interval (via the trace). Since the trace of a curve over $\mathbb{F}_q$ is the negative of the trace of its quadratic twist, and the quadratic twist is generally the only twist, we can use the $j$-invariant to uniquely identify elements of the isogeny class of trace $t \neq 0$ up to $\mathbb{F}_q$-isomorphism, even though $j$ normally only classifies curves up to $\overline{\mathbb{F}}_q$-isomorphism. We can handle $j = 0$ and 1728 as rare special cases, but for the case $t = 0$ we need to be more careful.

Now let us focus on a single $\mathbb{F}_q$-isogeny class. The isogeny class immediately breaks up into a union of $\mathbb{F}_q$-isomorphism classes. The modern way of looking at an $\mathbb{F}_q$-isogeny class is as a graph, with $\mathbb{F}_q$-isomorphism classes of curves for vertices, and $\mathbb{F}_q$-isomorphism classes of isogenies for edges (isogenies $\phi_1 : \mathcal{E}_1 \to \mathcal{E}_1'$ and $\phi_2 : \mathcal{E}_2 \to \mathcal{E}_2'$ are isomorphic if there are isomorphisms $\iota : \mathcal{E}_1 \to \mathcal{E}_2$ and $\mathcal{E}_1' \to \mathcal{E}_2'$ such that $\phi_2 \circ \iota = \iota' \circ \phi_1$).

Tate's theorem is not constructive, so we generally don't know how to get from one point to another in an isogeny graph. The difficulty of computing a path representing an unknown isogeny between given elliptic curves in the same isogeny class is the source of most hard problems in isogeny-based cryptography.

To take a closer look at the structure of isogeny graphs we need to classify isogenies, and to break them down to into fundamental components. Our main tool for this is the *degree*. Since an isogeny $\phi : \mathcal{E} \to \mathcal{E}'$ is defined by nonconstant rational maps, it induces an extension $\phi^{\#} : \mathbb{F}_q(\mathcal{E}') \hookrightarrow \mathbb{F}_q(\mathcal{E})$ of function fields; the degree $\deg(\phi)$ of $\phi$ is defined to be the degree of that extension. (We extend the definition of degree to homomorphisms by defining the degree of zero maps to be 0.) If $\phi : \mathcal{E} \to \mathcal{E}'$ and $\phi' : \mathcal{E}' \to \mathcal{E}''$ are isogenies, then $\deg(\phi' \circ \phi) = \deg \phi \cdot \deg \phi'$. Two examples are particularly important: $\deg[m] = m^2$, and $\deg \pi = q$. If $\deg \phi = d$, then we say that $\phi$ is a *$d$-isogeny*.

Another important quality of isogenies is *(in)separability*, which we define according to the (in)separability of the corresponding function field extension. For our purposes, the purely inseparable isogenies are all iterated compositions of $p$-powering $(x, y) \mapsto (x^p, y^p)$ (such as Frobenius); these can be factored out of any other isogeny, and then what remains is separable.

Suppose $S$ is a finite subgroup of $\mathcal{E}(\overline{\mathbb{F}}_q)$. Now $S$ must include 0, and it is also fixed by $[-1]$; so $S$ is determined precisely by the $x$-coordinates of its nonzero elements. We can therefore encode $S$ as a polynomial $F_S(X) = \prod_P (X - x(P))$, where the product runs over the nonzero points $P$ of $S$ in such a way that $P$ is included iff $-P$ is not. The subgroup $S$ is defined over $\mathbb{F}_q$ if and only if the polynomial $F_S$ has coefficients in $\mathbb{F}_q$.

Being homomorphisms, isogenies have kernels. The kernel of an isogeny $\phi : \mathcal{E} \to \mathcal{E}'$ is always a finite subgroup of $\mathcal{E}$. If $\phi$ is separable, then $\#\ker \phi = \deg \phi$. The points of $\ker \phi$ are generally defined over an extension of $\mathbb{F}_q$, but $\ker \phi$ can

be encoded as the *kernel polynomial* $F_{\ker \phi}$, which is defined over $\mathbb{F}_q$. Separable isogenies are defined by their kernels, up to isomorphism.

Going in the other direction, given a finite subgroup $S$ of $\mathcal{E}$ defined over $\mathbb{F}_q$, there exists a separable *quotient isogeny* $\phi : \mathcal{E} \to \mathcal{E}/S$ with $\ker \phi = S$. The isogeny and the curve $\mathcal{E}/S$ are both defined up to $\mathbb{F}_q$-isomorphism; they can be computed using *Vélu's formulæ* [137]. (If $S$ is encoded as the polynomial $F_S$, then we compute $\phi_S$ using the symmetric version of Vélu's formulæ in [86, §2.4].)

Given an ideal $\mathfrak{a} \subset \mathrm{End}(\mathcal{E})$, we can consider the subgroup $\mathcal{E}[\mathfrak{a}] := \cap_{\psi \in \mathfrak{a}} \ker \psi$. This is the kernel of an isogeny $\phi : \mathcal{E} \to \mathcal{E}/\mathcal{E}[\mathfrak{a}]$; the isogenies that arise in this way are central to the key exchange of §14. The degree of $\phi$ is the norm of $\mathfrak{a}$ in $\mathbb{Z} \subset \mathrm{End}(\mathcal{E})$. If $\mathfrak{a} = (\psi)$ is principal, then $\phi$ is isomorphic to $\psi$.

Given any $d$-isogeny $\phi : \mathcal{E} \to \mathcal{E}'$, we can compute the subgroup $S = \phi(\mathcal{E}[d]) \subset \mathcal{E}'$, and then the quotient $\phi_S : \mathcal{E}' \to \mathcal{E}'/S$ is a $d$-isogeny such that $\phi_S \circ \phi$ has kernel $\mathcal{E}[d]$; hence, $\phi_S$ is isomorphic to a $d$-isogeny $\phi^\dagger : \mathcal{E}' \to \mathcal{E}$ such that $\phi^\dagger \circ \phi = [d]$ on $\mathcal{E}$ (and $\phi \circ \phi^\dagger = [d]$ on $\mathcal{E}'$). We call $\phi^\dagger$ the *dual* of $\phi$. The upshot is that every $d$-isogeny $\mathcal{E} \to \mathcal{E}'$ has a corresponding $d$-isogeny $\mathcal{E}' \to \mathcal{E}$.

Every isogeny can be factored into a composition of isogenies of prime degree, though there are two important caveats: factorization is not unique, and generally a factorization may only exist over some extension field. For example, if $\ell \neq p$ is prime, then $\mathcal{E}(\overline{\mathbb{F}}_q) \cong (\mathbb{Z}/\ell\mathbb{Z})^2$, so there are $\ell + 1$ order-$\ell$ subgroups $S \subset \mathcal{E}(\overline{\mathbb{F}}_q)[\ell]$, each the kernel of a different isogeny $\phi_S : \mathcal{E} \to \mathcal{E}/S$, and then the dual isogeny gives us a factorization $[\ell] = {\phi_S}^\dagger \circ \phi_S$. Each decomposition is only defined over the field of definition of the associated subgroup $S$.

Just as we decompose isogenies into $\ell$-isogenies, so consider the subgraphs formed by $\ell$-isogenies. The structures of $\ell$-isogeny graphs depend strongly on the endomorphism rings of curves in the isogeny class, as we will see.

A curve $\mathcal{E}$ is *supersingular* if $p$ divides its trace (over $\mathbb{F}_p$, this implies the trace is 0). If $\mathcal{E}$ is not supersingular, then it is *ordinary*. The $j$-invariant of any supersingular curve is in $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$, so any supersingular curve is isomorphic to one defined over $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$. There are roughly $\lfloor p/12 \rfloor$ supersingular $j$-invariants in $\mathbb{F}_{p^2}$, of which $O(\sqrt{p})$ are in $\mathbb{F}_p$ (more precisely, this number is the class number of $\mathbb{Q}(\sqrt{-p})$). Since supersingularity is defined by the trace, either all of the curves in an isogeny class are supersingular, or all of them are ordinary; the two kinds of curves do not mix.

There are two possibilities for the general structure of the endomorphism ring of an elliptic curve over a finite field:

**commutative** $\mathrm{End}(\mathcal{E})$ is isomorphic to an order in a quadratic imaginary field; or

**noncommutative** $\mathrm{End}(\mathcal{E})$ is isomorphic to a maximal order in a quaternion algebra.

All ordinary curves have commutative endomorphism rings. If a supersingular curve is defined over $\mathbb{F}_p$, then its endomorphism ring is commutative[16]; if it is defined over $\mathbb{F}_{p^2}$, then its endomorphism ring is noncommutative.

[16] If we consider endomorphisms defined over $\mathbb{F}_{p^2}$, then the ring is noncommutative.

The commutative case is relatively simple: each $\mathrm{End}(\mathcal{E})$ is an order in $K = \mathbb{Q}(\pi)$ containing the quadratic ring $\mathbb{Z}[\pi]$. The discriminant of $\mathbb{Z}[\pi]$ is $\Delta_\pi := t^2 - 4q = m^2\Delta_K$, where $\Delta_K$ is the fundamental discriminant of $K$. The algorithmic exploration of ordinary isogeny graphs begins with Kohel's thesis [86, Chapter 4]; these graphs are now mainstream computational tools in the arithmetic of elliptic curves and elliptic curve cryptography [66,61,67,81]. The analogous theory for supersingular curves over $\mathbb{F}_p$, whose endomorphism rings are commutative and thus behave like ordinary curves, was explored by Delfs and Galbraith [48].

If $\phi : \mathcal{E} \to \mathcal{E}'$ is an $\ell$-isogeny of endomorphism rings with commutative endomorphism rings (with $\ell$ prime), then there are three possibilities: $\mathrm{End}(\mathcal{E}) \cong \mathrm{End}(\mathcal{E}')$ (we say $\phi$ is *horizontal*), $\mathrm{End}(\mathcal{E}) \subset \mathrm{End}(\mathcal{E}')$ with index $\ell$ (we say $\phi$ is *ascending*), or $\mathrm{End}(\mathcal{E}) \supset \mathrm{End}(\mathcal{E}')$ with index $\ell$ (we say $\phi$ is *descending*). An $\ell$-isogeny can only be ascending or descending if $\ell$ divides the conductor $m$ of $\mathbb{Z}[\pi]$ in $O_K$, and an $\ell$-isogeny $\phi : \mathcal{E} \to \mathcal{E}'$ can only be horizontal if $\mathrm{End}(\mathcal{E})$ and $\mathrm{End}(\mathcal{E}')$ are locally isomorphic to the maximal order $O_K$ of $K$ at $\ell$: that is, if $\mathrm{End}(\mathcal{E}) \otimes \mathbb{Z}_\ell \cong \mathrm{End}(\mathcal{E}') \otimes \mathbb{Z}_\ell \cong O_K \otimes \mathbb{Z}_\ell$. The $\ell$-isogenies of ordinary curves thus form "volcano" structures: cycles of horizontal isogenies link the curves $\mathcal{E}$ with $\mathrm{End}(\mathcal{E}) \cong O_K$, and from each of these curves a regular tree grows downwards, with its leaves in the curves with $\mathrm{End}(\mathcal{E}) \cong \mathbb{Z}[\pi]$ (which is minimal). The vertices with $\mathrm{End}(\mathcal{E}) \cong O_K$ have two horizontal $\ell$-isogenies (or one or zero, if the cycle is degenerate), and $\ell - 1$ descending isogenies; each other vertex has one ascending and $\ell$ descending isogenies, except for the minimal vertices, which have no further descending isogenies.

From our perspective, what is most interesting about the commutative case is that the (isomorphism classes) of curves $\mathcal{E}$ with $\mathrm{End}(\mathcal{E}) \cong O_K$ form a PHS under the action of the class group of $\mathrm{Cl}(O_K)$. We met this PHS in Example 4.

The noncommutative case is much more complicated, and we will be much more brief here. The algorithmic applications of the full supersingular isogeny graph go back to Mestre and Oesterlé [101], and more detail appears in the second half of Kohel's thesis [86, Chapter 7]. In the non-commutative case, the $\ell$-isogeny graph is $(\ell + 1)$-regular and connected, and it is an expander graph.

## 14 Commutative isogeny-based key exchange

Recall the PHS space from Example 4, which Couveignes conjectured was an HHS: fix a prime power $q$ and an integer $t$ with $|t| \leq 2\sqrt{q}$, set $\Delta := t^2 - 4q$, and let $\mathcal{O}_K$ be the maximal order (the ring of integers) of the quadratic imaginary field $K := \mathbb{Q}(\sqrt{\Delta})$. For this HHS,

- the space $\mathcal{X}$ is the set of isomorphism classes of elliptic curves over $\mathbb{F}_q$ of trace $t$ whose endomorphism rings are isomorphic to $\mathcal{O}_K$;
- the group $\mathfrak{G}$ is the ideal class group $\mathrm{Cl}(\mathcal{O}_K)$ of $\mathcal{O}_K$; and
- the action $(\mathfrak{a}, \mathcal{E}) \mapsto \mathfrak{a} \cdot \mathcal{E}$ is evaluated by computing the isogeny $\phi : \mathcal{E} \to \mathcal{E}/\mathcal{E}[\mathfrak{a}]$, and taking $\mathfrak{a} \cdot \mathcal{E}$ to be the isomorphism class of $\mathcal{E}/\mathcal{E}[\mathfrak{a}]$.

The cardinality $N$ of $\mathfrak{G}$ is the class number of $O_K$, which is roughly $\sqrt{\Delta_K}$, where $\Delta_K$ is the discriminant of $K$ (essentially the squarefree part of $\Delta$). There is no

point in not maximising $N$ with respect to $q$, so we should use $t$ such that $\Delta$ is already a fundamental discriminant; this forces all curves in the isogeny class to have $\mathbb{Z}[\pi] = \mathrm{End}(\mathcal{E}) = O_K$, and then $N = \#\mathfrak{G} \sim \sqrt{q}$.

The vectorization and parallelization problems in this HHS are expressed concretely in terms of computing paths in isogeny graphs. The fastest known classical algorithms for vectorization and parallelization in this HHS are the generic square-root algorithms, which run in time $O(\sqrt[4]{q})$. In the quantum world, Childs, Jao, and Soukharev have defined a subexponential quantum isogeny evaluation algorithm [36], which in combination with Kuperberg's algorithm gives a full subexponential quantum algorithm for solving vectorization in this HHS. This applies identically to the ordinary and commutative-supersingular cases. Further analysis of this approach can be found in [22].

Couveignes defined a key exchange (essentially Algorithms 7 and 8) and an identification protocol in this HHS in [42]. These protocols were essentially unknown outside the French community until Rostovtsev and Stolbunov independently proposed a public key encryption scheme based on the same HHS [120]. Stolbunov [129] then derived more protocols, including an interactive key exchange scheme similar to Algorithms 7 and 8. The only real difference between Couveignes' and Stolbunov's cryptosystems is in the sampling of private keys, each representing one of the two approaches mentioned in §10. Couveignes uses a true uniform random sampling over the whole of the keyspace, then applies a lattice reduction-based algorithm to produce an equivalent key whose action is efficiently computable. Rostovtsev and Stolbunov sample keys from a subset of efficiently computable keys whose distribution they conjecture to be close enough to the uniform distribution on the entire group.

One particularly nice aspect of these schemes is that key validation can be made simple and efficient (see [46, §5.4]), so we can safely use the scheme for static key exchange. Since the group action is simple and transitive, every element of the space is a legitimate public key. To validate a given $x$ in $\mathbb{F}_q$ as a public key, therefore, it suffices to check that $x$ is the $j$-invariant of a curve with endomorphism ring $O_K$. We immediately construct a curve $\mathcal{E}$ with $j$-invariant $x$, and check that it has the right trace (which amounts to checking that $\mathcal{E}(\mathbb{F}_q)$ has the claimed cardinality), switching to the quadratic twist if necessary. This ensures that $\mathrm{End}(\mathcal{E}) \subseteq O_K$; if $t$ is chosen such that $\mathbb{Z}[\pi] = O_K$, then we are already done; otherwise, we check $\mathrm{End}(\mathcal{E}) = O_K$ using Kohel's algorithm [86].

Regardless of how the private key ideals are sampled, by the time we want to use them in the group action they are presented as factored ideals

$$\mathfrak{a} = \prod_{i=1}^{r} \mathfrak{l}_i^{e_i} \quad \text{with} \quad -B_i \leq e_i \leq B_i \,,$$

where the $\mathfrak{l}_i$ are distinguished prime ideals whose corresponding $\ell_i$-isogenies can be evaluated very quickly. If the cost of evaluating an isogeny associated with kernel $\mathcal{E}[\mathfrak{l}_i]$ (for a random $\mathcal{E}$ in the isogeny class) is $C_i$, then the exponent bounds $B_i$ should be chosen in such a way that the cost of evaluation $\sum_{i=1}^{r} B_i C_i$ is minimised while keeping the number of private keys $\prod_{i=1}^{r}(2B_i + 1)$ big enough.

Suppose then that want to compute an $\ell$-isogeny from an elliptic curve $\mathcal{E}/\mathbb{F}_q$ for some prime $\ell \neq p$. We consider two methods of computing $\ell$-isogenies here. The classic approach is based on modular polynomials. An alternative approach based on Vélu's formulæ was originally proposed in [46], and subsequently used in [34]. Both approaches are discussed in greater detail in [46, §3.2].

First, consider the "modular" approach. Recall that the $\ell$-th (classical[17]) modular polynomial $\Phi_\ell(J_1, J_2)$ is defined over $\mathbb{Z}$, is monic of degree $\ell + 1$ in both $J_1$ and $J_2$, and the roots of $\Phi_\ell(j(\mathcal{E}), X)$ in $\mathbb{F}_q$ are the $j$-invariants of the curves $\ell$-isogenous to $\mathcal{E}$ over $\mathbb{F}_q$. In fact, the $\mathbb{F}_q$-irreducible factors correspond to Galois orbits of $\ell$-isogenies (or, equivalently, to Galois orbits of order-$\ell$ subgroups of $\mathcal{E}$). To compute the $\ell$-isogenous curves up to isomorphism, therefore, we (pre)compute $\Phi_\ell$ and reduce it modulo $p$; then we evaluate one variable at $j(\mathcal{E})$, and compute the roots in $\mathbb{F}_q$ of the resulting univariate polynomial.

There are $\ell + 1$ curves $\ell$-isogenous to $\mathcal{E}$ over $\overline{\mathbb{F}}_q$. Of these isogenies, at most two can preserve the endomorphism ring. If we choose $\mathcal{E}$ such that $\mathrm{End}(\mathcal{E})$ is the maximal order and equal to $\mathbb{Z}[\pi]$ (or at least such that $\ell$ does not divide the conductor of $\mathbb{Z}[\pi]$ in $O_K$, so $\mathbb{Z}[\pi]$ is locally maximal) then $\Phi_\ell(j(\mathcal{E}), X)$ will have only two roots, corresponding precisely to these horizontal isogenies. If this is the first step in a walk of $\ell$-isogenies then we must determine which of the two is in the "correct" direction, corresponding to the ideal; we can do this by checking the eigenvalue of Frobenius on the kernel, for example. But if we have already started walking, then there is no need to do this: we know that the "wrong" isogeny is the dual of the preceding step, so we just ignore the $j$-invariant of that curve. The total cost of this approach is dominated by finding the roots of $\Phi_\ell(j(\mathcal{E}), \mathbb{F}_q)$, which is $O(\ell \log q)$ $\mathbb{F}_q$-operations.

The alternative "Vélu" approach is to construct isogeny kernels explicitly, and compute the corresponding isogeny steps using Vélu's formulæ [137]. The idea is simple: suppose $\mathcal{E}(\mathbb{F}_q)$ contains a point $P_\ell$ of order $\ell$; we want to compute the isogeny $\mathcal{E} \to \mathcal{E}/\langle P_\ell \rangle$ with kernel generated by $P_\ell$. We can compute the kernel polynomial $F(X) = \prod_{i=1}^{(\ell-1)/2}(X - [i]P_\ell)$ in $\widetilde{O}(\ell)$ $\mathbb{F}_q$-operations; if $P_\ell$ is defined over a small extension of $\mathbb{F}_q$, say $\mathbb{F}_{q^k}$, then the cost is $\widetilde{O}(k^2\ell)$ $\mathbb{F}_q$-operations. We can then apply Vélu's formulæ (as in [137] or [86, §2.4]) to compute an equation for $\mathcal{E}/\langle P_\ell \rangle$; we do not need an expression for the isogeny itself. The total cost is dominated by the cost of computing $F$, which is $\widetilde{O}(k^2\ell)$ $\mathbb{F}_q$-operations.

The Vélu approach is much faster than the modular approach when $k^2 \ll \log q$, but it requires us to use isogeny classes of curves with many small-order subgroups over very low-degree extensions. Such curves are rare, and hard to find by exhaustive search: constructing them presents similar challenges to the construction of pairing-friendly curves (though here we want many small primes dividing the order over a degree-$k$ extension, rather than one big prime). We might try to do better by using the CM method [3,130], which constructs elliptic curves with a specified group order—but the CM method only works when the

[17] We use classical modular polynomials here for simplicity, but alternative modular polynomials such as Atkin's, which have smaller degree, are better in practice. These degrees are still in $O(\ell)$, so the asymptotic efficiency of this approach does not change.

discriminant $\Delta_K$ of the maximal order (and hence the class group of the maximal order) is very small, because $\#\mathrm{Cl}(O_K) \sim \sqrt{|-\Delta_K|}$. This means that if we use the CM method to generate parameters, then the private key space is far too small for these cryptosystems to be secure. In [46] curve parameters are selected by running an extensive search to maximise the number of primes $\ell$ with points in $\mathcal{E}[\ell](\mathbb{F}_{q^k})$ for smallish $k$, using the Vélu approach for these primes and the modular approach for the others. This gives a significant improvement over the pure modular approach, but the result is still far from truly practical.

CSIDH [34] steps around this obstruction in an extremely neat way, by switching to supersingular curves over $\mathbb{F}_p$. Since their endomorphism rings are commutative quadratic orders, these curves behave like ordinary curves, and the Couveignes–Rostovtsev–Stolbunov protocol carries over without modification. However, the fact that these curves necessarily have order $p+1$ makes it extremely simple to control their group structure and class group size by appropriately choosing $p$ from within the desired range. This close control means that we can force all of the small primes to be "Vélu" with $k = 1$, which results in a speedup that beats ordinary-curve constructions like that of [46] by orders of magnitude. Key validation is also simpler for these curves. We are unaware of any impact on security, negative or positive, stemming from the use of supersingular curves as opposed to ordinary curves; so far, each attack described as targeting either CRS or CSIDH (e.g. [22]) applies equally to the other. CSIDH therefore represents a practical post-quantum Diffie–Hellman replacement, though the development of efficient side-channel-aware implementations of commutative isogeny protocols remains an open problem.

## 15 Supersingular isogeny Diffie–Hellman

We conclude with a brief discussion of Jao and De Feo's supersingular isogeny Diffie–Hellman, known as SIDH [80,45]. On the surface, SIDH resembles the commutative isogeny key exchange of §14: Alice and Bob each compute a sequence of isogenies to arrive at their public keys, and later the shared secret. However, the differences are striking.

The most fundamental difference is that the endomorphism rings in SIDH are noncommutative, so the algebraic objects acting on the isogeny class are not abelian groups: SIDH falls squarely outside the HHS framework. In particular, Alice and Bob's isogeny walks do not automatically commute; some extra data must be passed around to correctly orient their walks for the second phase of the key exchange.

The second crucial difference with commutative isogeny key exchange is that the underlying $\ell$-isogeny graphs are no longer cycles; rather, each is $(\ell+1)$-regular, connected, and an expander graph. Since there are $\Theta(p)$ vertices, computing random sequences of $O(\log p)$ $\ell$-isogenies from a given base curve takes us to a distribution of curves that we expect to be close to a uniform random distribution on the isogeny class.

To define the protocol, we fix distinct primes $\ell_A$ and $\ell_B$ (these will be very small, typically 2 and 3), and exponents $n_A$ and $n_B$, respectively, and let $p$ be a prime such that $p = c \cdot \ell_A^{n_A} \cdot \ell_B^{n_B} \pm 1$ for some very small $c$. We want to choose $\ell_A$ and $\ell_B$ such that $\ell_A^{n_A}$ and $\ell_B^{n_B}$ are roughly the same size; ideally, $\ell_A^{n_A} \sim \ell_B^{n_B} \sim \sqrt{p}$.

Now consider the supersingular isogeny class over $\mathbb{F}_{p^2}$: every curve $\mathcal{E}$ in it has $\mathcal{E}[\ell_A^{n_A}] \cong (\mathbb{Z}/\ell_A^{n_A}\mathbb{Z})^2$ and $\mathcal{E}[\ell_B^{n_B}] \cong (\mathbb{Z}/\ell_B^{n_B}\mathbb{Z})^2$. Fix a base curve $\mathcal{E}_0$ in the isogeny class, along with bases $(P_A, Q_A)$ of $\mathcal{E}_0[\ell_A^{n_A}]$ and $(P_B, Q_B)$ of $\mathcal{E}_0[\ell_B^{n_B}]$.

First, key generation. Alice samples a random $a$ in $\mathbb{Z}/\ell_A^{n_A}\mathbb{Z}$ as her private key; the point $P_A + [a]Q_A$ has exact order $\ell_A^{n_A}$, and generates the kernel of an $\ell_A^{n_A}$-isogeny $\phi_A : \mathcal{E}_0 \to \mathcal{E}_A \cong \mathcal{E}_0/\langle P_A + [a]Q_A \rangle$, which she computes as a series of $\ell_A$-isogenies. Her public key is $(\mathcal{E}_A, \phi_A(P_B), \phi_A(Q_B))$. Bob samples a private key $b$ in $\mathbb{Z}/\ell_B^{n_B}\mathbb{Z}$ and computes the $\ell_B^{n_B}$-isogeny $\phi_B : \mathcal{E}_0 \to \mathcal{E}_B \cong \mathcal{E}_0/\langle P_B + [b]Q_B \rangle$ as a series of $\ell_B$-isogenies; his public key is $(\mathcal{E}_B, \phi_B(P_A), \phi_B(Q_A))$. There is plenty of redundant information in these public keys, and they can be compressed following the suggestions in [39].

To complete the key exchange, Alice computes the $\ell_A^{n_A}$-isogeny $\phi'_A : \mathcal{E}_B \to \mathcal{E}_{BA} = \mathcal{E}_B/\langle \phi_B(P_A) + [a]\phi_B(Q_A) \rangle$, and Bob computes the $\ell_B^{n_B}$-isogeny $\phi'_B : \mathcal{E}_A \to \mathcal{E}_{AB} = \mathcal{E}_A/\langle \phi_A(P_B) + [b]\phi_A(Q_B) \rangle$. The shared secret is the $j$-invariant $j(\mathcal{E}_{AB}) = j(\mathcal{E}_{BA})$ in $\mathbb{F}_{p^2}$; the curves $\mathcal{E}_{AB}$ and $\mathcal{E}_{BA}$ have the same $j$-invariant because both are isomorphic to $\mathcal{E}_0/\langle P_A + [a]Q_A + P_B + [b]Q_B \rangle$. The relationships between these curves is illustrated in Figure 2.

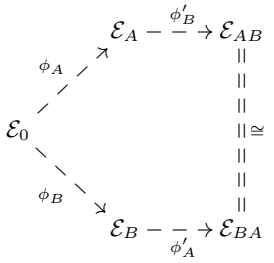


**Fig. 2.** Supersingular Isogeny Diffie–Hellman.

Jao, De Feo, and Plût specified efficient algorithms for SIDH in [45]. The first competitive public implementation was due to Costello, Longa, and Naehrig [40]. A lot of effort has since been put into improving the algorithmic and space efficiency of SIDH [39,38,60], and optimizing arithmetic in its specialized finite fields [25,24]. One particlarly nice feature of SIDH in comparison to commutative isogeny DH is that the isogenies in SIDH all have degree either $\ell_A$ or $\ell_B$; both are fixed, and typically tiny, so computing the individual isogeny steps is much faster in the supersingular protocol, and requires much less code and precomputation.

Recovering the private key from an SIDH public key—a noncommutative analogue of vectorization—amounts to computing an isogeny between the base

curve $\mathcal{E}_0$ and the target public key curve $\mathcal{E}_A$. We can do this using an algorithm due to Delfs and Galbraith [48], inspired by an algorithm for the ordinary case due to Galbraith, Hess, and Smart [67]. The algorithm walks randomly through the supersingular isogeny graph from the starting and ending curves, until curves defined over $\mathbb{F}_p$ are detected—and then finding a path between those two curves, to complete the desired isogeny, is analogous to breaking a CSIDH key (though with completely different security parameters). Alternatively, Adj, Cervantes–Vázquez, Chi–Dominguez, Menezes, and Rodríguez–Henríquez have given a useful analysis of the van Oorschot–Wiener algorithm applied to this problem [2]. The asymptotic cost of either approach is in $O(\sqrt[4]{p})$ $\mathbb{F}_{p^2}$-operations if $\ell_A^{n_A} \sim \ell_B^{n_B} \sim \sqrt{p}$. In the quantum setting, we can apply Tani's claw-finding algorithm [132] to find a curve in the intersection of the sets of curves $\ell_A^{n_A/2}$-isogenous to $\mathcal{E}_0$ and $\mathcal{E}_A$ with a query complexity of $O(\ell_A^{n_A/3})$ (or we can attack Bob's public key in $O(\ell_B^{n_B/3})$), which is $O(p^{1/6})$ when $\ell_A^{n_A} \sim \ell_B^{n_B} \sim \sqrt{p}$. The fact that the subexponential Childs–Jao–Soukharev algorithm does not apply in the noncommutative case was one of the motivations for developing SIDH.

But SIDH keys do not simply present the target curve of an unknown isogeny: they also present images of distinguished torsion bases, which may help cryptanalysis [112]. The precise nature of the cryptographic problems underlying SIDH is quite complicated, but Urbanik and Jao's survey of these problems provides useful analysis [134], while Eisentraeger, Hallgren, Lauter, Morrison, and Petit go further into the connections with the endomorphism ring [56].

Finding an isogeny between two supersingular curves over $\mathbb{F}_{p^2}$ is equivalent to determining their endomorphism rings, under reasonable heuristics [56,86,87]. This makes an interesting contrast with the commutative case, where the endomorphism rings are presumed known, and in any case can be computed using Kohel's algorithm [86]. As we have seen, determining the endomorphism ring is an important step in public key validation in commutative isogeny key exchange.

Key validation is especially problematic for SIDH. Suppose we have an algorithm which, given a prime $\ell$, a positive integer $n$, and a curve $\mathcal{E}$, efficiently decides whether $\mathcal{E}$ is $\ell^n$-isogenous to $\mathcal{E}_0$. Such an algorithm would allow us to verify whether Alice or Bob's public key was honestly generated (by calling the algorithm on $(\ell_A, n_A, \mathcal{E}_A)$ or $(\ell_B, n_B, \mathcal{E}_B)$, respectively). However, as we see in [70, §6.2] and [133], this algorithm can also be used to efficiently recover secret keys from public keys. Indeed, take Alice's public curve $\mathcal{E}_A$; there are $\ell_A + 1$ curves $\ell_A$-isogenous to it. Computing each of these isogenies $\phi : \mathcal{E}_A \to \mathcal{E}'_A$, we call the algorithm on $(\ell_A, n_A - 1, \mathcal{E}_0, \mathcal{E}'_A)$; if it returns true, then $\phi$ is the last $\ell_A$-isogeny in Alice's secret key. Iterating this procedure reveals the entire key.

Problematic key validation makes defining a CCA-secure SIDH-based KEM more complicated than the equivalent in the commutative case. SIKE [9], which is the only isogeny-based candidate KEM in the NIST process, handles this by applying the Hofheinz–Hövelmanns–Kiltz a variant of the Fujisaki–Okamoto transform [78,65] to SIDH; this entails a nontrivial performance hit.

On a formal level, there are some profound differences between SIDH and classical Diffie–Hellman. The most obvious is the lack of symmetry in SIDH be-

tween Alice and Bob, whose roles are no longer interchangeable. This is reflected by their distinct and incompatible key spaces, which are in turn distinct from the shared secret space and the space the base curve lives in. Alice's private key encodes a sequence of $\ell_A$-isogenies of length $n_A$, while Bob's encodes a sequence of $\ell_B$-isogenies of length $n_B$. Alice's public key belongs to the space of (isomorphism classes of) elliptic curves equipped with a distinguished $\ell_B^{n_B}$-torsion basis, while Bob's is equipped with an $\ell_A^{n_A}$-torsion basis instead. The base curve $\mathcal{E}_0$ is drawn from yet another space: it is equipped with an $\ell_A^{n_A}\ell_B^{n_B}$-torsion basis.

This asymmetry might seem like a curious but minor inconvenience: the participants just need to decide who is Alice and who is Bob before each key exchange. More importantly, though, this asymmetry is incompatible with most of the theoretical machinery that we use to reason about Diffie–Hellman and its hardness. We have already seen how group Diffie–Hellman oracles create black-box field structures on prime-order groups, while HHS Diffie–Hellman oracles create a black-box group structures. In SIDH, however, a Diffie–Hellman oracle defines no binary operation on any set, let alone an interesting algebraic structure. This plurality of spaces makes it hard to adapt hidden-number-problem-style arguments [21,5] for hardcore bits to the SIDH context in a natural way, though a valiant effort has been made by Galbraith, Petit, Shani, and Ti [68].

*Remark 3.* At first glance, the fact that SIKE is the only isogeny-based KEM submitted to the NIST post-quantum process, competing with 58 others mostly based on codes, lattices, and polynomial systems, might suggest that it is a strange outlier. However, this uniqueness is not so much an indicator of lack of support, so much as a sign of rare convergence and consensus in the elliptic-curve cryptography community—convergence that did not occur to the same extent in the communities working on other post-quantum paradigms. The fact that there was only one isogeny-based submission reflects the general agreement that this was the right way to do isogeny-based key agreement at that point in time. The more flexible CSIDH scheme was not developed until later, when the NIST process was already underway, and so it was not part of the conversation.

## References

1. M. Abdalla, M. Bellare, and P. Rogaway. DHAES: an encryption scheme based on the Diffie–Hellman problem. *IACR Cryptology ePrint Archive*, 1999:7, 1999.
2. G. Adj, D. Cervantes-Vázquez, J. Chi-Domínguez, A. Menezes, and F. Rodríguez-Henríquez. On the cost of computing isogenies between supersingular elliptic curves. *IACR Cryptology ePrint Archive*, 2018:313, 2018.
3. A. Agashe, K. E. Lauter, and R. Venkatesan. Constructing elliptic curves with a known number of points over a prime field. In van der Poorten and Stein [136], pages 1–17.
4. C. Aguilar, P. Gaborit, P. Lacharme, J. Schrek, and G. Zémor. Noisy Diffie–Hellman protocols, 2010. Slides presented at the recent results session of PQC 2010, available at `https://pqc2010.cased.de/rr/03.pdf`.

5. A. Akavia. Solving hidden number problem with one bit oracle and advice. In S. Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 337–354. Springer, 2009.
6. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange – A new hope. In T. Holz and S. Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343. USENIX Association, 2016.
7. A. Antipa, D. R. L. Brown, A. Menezes, R. Struik, and S. A. Vanstone. Validation of elliptic curve public keys. In Y. Desmedt, editor, *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*, pages 211–223. Springer, 2003.
8. F. Armknecht, T. Gagliardoni, S. Katzenbeisser, and A. Peter. General impossibility of group homomorphic encryption in the quantum world. In H. Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*, pages 556–573. Springer, 2014.
9. R. Azarderakhsh, B. Koziel, M. Campagna, B. LaMacchia, C. Costello, P. Longa, L. De Feo, M. Naehrig, B. Hess, J. Renes, A. Jalali, V. Soukharev, D. Jao, and D. Urbanik. Supersingular isogeny key encapsulation, 2017.
10. R. Balasubramanian and N. Koblitz. The improbability that an elliptic curve has subexponential discrete log problem under the Menezes–Okamoto–Vanstone algorithm. *J. Cryptology*, 11(2):141–145, 1998.
11. R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In P. Q. Nguyen and E. Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2014.
12. J. Benaloh. Simple verifiable elections. In D. S. Wallach and R. L. Rivest, editors, *2006 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'06, Vancouver, BC, Canada, August 1, 2006*. USENIX Association, 2006.
13. K. Bentahar. The equivalence between the DHP and DLP for elliptic curves used in practical applications, revisited. In N. P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*, pages 376–391. Springer, 2005.
14. D. J. Bernstein. Curve25519: New Diffie–Hellman speed records. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006.
15. D. J. Bernstein. Differential addition chains. Preprint, 2006.
16. D. J. Bernstein, C. Chuengsatiansup, T. Lange, and P. Schwabe. Kummer strikes back: New DH speed records. In P. Sarkar and T. Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan,*

*R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 317–337. Springer, 2014.
17. D. J. Bernstein, S. Engels, T. Lange, R. Niederhagen, C. Paar, P. Schwabe, and R. Zimmermann. Faster discrete logarithms on FPGAs, 2016. Document ID: 01ac92080664fb3a778a430e028e55c8.
18. D. J. Bernstein, T. Lange, and P. Schwabe. On the correct use of the negation map in the Pollard rho method. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, volume 6571 of *Lecture Notes in Computer Science*, pages 128–146. Springer, 2011.
19. D. Boneh. The decision Diffie–Hellman problem. In J. Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 1998.
20. D. Boneh and R. J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In Koblitz [85], pages 283–297.
21. D. Boneh and R. Venkatesan. Hardness of computing the most significant bits of secret keys in Diffie–Hellman and related schemes. In Koblitz [85], pages 129–142.
22. X. Bonnetain and A. Schrottenloher. Quantum security analysis of CSIDH and ordinary isogeny-based schemes. *IACR Cryptology ePrint Archive*, 2018:537, 2018.
23. J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 553–570. IEEE Computer Society, 2015.
24. J. W. Bos and S. Friedberger. Fast arithmetic modulo $2^x p^y \pm 1$. In N. Burgess, J. D. Bruguera, and F. de Dinechin, editors, *IEEE Symposium on Computer Arithmetic – ARITH 2017*, pages 148–155. IEEE Computer Society, 2017.
25. J. W. Bos and S. Friedberger. Arithmetic considerations for isogeny based cryptography. *IACR Cryptology ePrint Archive*, 2018:376, 2018.
26. G. Brassard, editor. *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*. Springer, 1990.
27. R. Bröker, K. E. Lauter, and A. V. Sutherland. Modular polynomials via isogeny volcanoes. *Math. Comput.*, 81(278):1201–1231, 2012.
28. D. R. L. Brown. CM55: special prime-field elliptic curves almost optimizing den Boer's reduction between Diffie–Hellman and discrete logs. *IACR Cryptology ePrint Archive*, 2014:877, 2014.
29. J. Buchmann, R. Scheidler, and H. C. Williams. A key-exchange protocol using real quadratic fields. *Journal of Cryptology*, 7:171–199, 1994.
30. J. Buchmann, T. Takagi, and U. Vollmer. Number field cryptography. In van der Poorten and Stein [136], pages 111–125.
31. J. A. Buchmann and H. C. Williams. A key exchange system based on real quadratic fields. In Brassard [26], pages 335–343.
32. R. Canetti and H. Krawczyk. Analysis of Key-Exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*. Springer, 2001.
33. J. W. S. Cassels. *Lectures on Elliptic Curves*, volume 24 of *London Mathematical Society Student Texts*. Cambridge University Press, 1991.

34. W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. CSIDH: an efficient post-quantum commutative group action. *IACR Cryptology ePrint Archive*, 2018:383, 2018.
35. J. H. Cheon. Security analysis of the strong Diffie–Hellman problem. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 2006.
36. A. Childs, D. Jao, and V. Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *Journal of Mathematical Cryptology*, 8(1):1–29, 2014.
37. J. Coron and J. B. Nielsen, editors. *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, 2017.
38. C. Costello and H. Hisil. A simple and compact algorithm for SIDH with arbitrary degree isogenies. In Takagi and Peyrin [131], pages 303–329.
39. C. Costello, D. Jao, P. Longa, M. Naehrig, J. Renes, and D. Urbanik. Efficient compression of SIDH public keys. In Coron and Nielsen [37], pages 679–706.
40. C. Costello, P. Longa, and M. Naehrig. Efficient algorithms for supersingular isogeny Diffie–Hellman. In M. Robshaw and J. Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 572–601. Springer, 2016.
41. C. Costello and B. Smith. Montgomery curves and their arithmetic. *Journal of Cryptographic Engineering*, Mar 2017.
42. J. M. Couveignes. Hard homogeneous spaces. *IACR Cryptology ePrint Archive*, 2006:291, 2006.
43. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003.
44. L. De Feo. Mathematics of isogeny based cryptography. *CoRR*, abs/1711.04062, 2017.
45. L. De Feo, D. Jao, and J. Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2014.
46. L. De Feo, J. Kieffer, and B. Smith. Towards practical key exchange from ordinary isogeny graphs. *IACR Cryptology ePrint Archive*, 2018:485, 2018.
47. I. Déchène. On the security of generalized Jacobian cryptosystems. *Adv. in Math. of Comm.*, 1(4):413–426, 2007.
48. C. Delfs and S. D. Galbraith. Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. *Des. Codes Cryptography*, 78(2):425–440, 2016.
49. B. den Boer. Diffie–Hellman is as strong as discrete log for certain primes. In S. Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 530–539. Springer, 1988.
50. J. Deneuville, P. Gaborit, and G. Zémor. Ouroboros: A simple, secure and efficient key exchange protocol based on coding theory. In T. Lange and T. Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto*

*2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*, volume 10346 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2017.
51. C. Diem and E. Thomé. Index calculus in class groups of non-hyperelliptic curves of genus three. *J. Cryptology*, 21(4):593–611, 2008.
52. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.
53. J. Ding. New cryptographic constructions using generalized learning with errors problem. *IACR Cryptology ePrint Archive*, 2012:387, 2012.
54. J. Ding, X. Xîe, and X. Lin. A simple provably secure key exchange scheme based on the learning with errors problem. *IACR Cryptology ePrint Archive*, 2012:688, 2012.
55. I. M. Duursma, P. Gaudry, and F. Morain. Speeding up the discrete log computation on curves with automorphisms. In K. Lam, E. Okamoto, and C. Xing, editors, *Advances in Cryptology - ASIACRYPT '99, International Conference on the Theory and Applications of Cryptology and Information Security, Singapore, November 14-18, 1999, Proceedings*, volume 1716 of *Lecture Notes in Computer Science*, pages 103–121. Springer, 1999.
56. K. Eisenträger, S. Hallgren, K. E. Lauter, T. Morrison, and C. Petit. Supersingular isogeny graphs and endomorphism rings: Reductions and solutions. In J. B. Nielsen and V. Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 329–368. Springer, 2018.
57. N. El Mrabet and M. Joye, editors. *Guide to Pairing-Based Cryptography*. Chapman and Hall/CRC, New York, 2016.
58. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
59. A. Enge, P. Gaudry, and E. Thomé. An $L(1/3)$ discrete logarithm algorithm for low degree curves. *J. Cryptology*, 24(1):24–41, 2011.
60. A. Faz-Hernández, J. López, E. Ochoa-Jiménez, and F. Rodríguez-Henríquez. A faster software implementation of the supersingular isogeny Diffie–Hellman key exchange protocol. *IEEE Transactions on Computers*, PP(99):1–1, 2017.
61. M. Fouquet and F. Morain. Isogeny volcanoes and the SEA algorithm. In C. Fieker and D. R. Kohel, editors, *Algorithmic Number Theory Symposium*, volume 2369 of *Lecture Notes in Computer Science*, pages 47–62, Berlin, Heidelberg, 2002. Springer Berlin / Heidelberg.
62. E. S. V. Freire, D. Hofheinz, E. Kiltz, and K. G. Paterson. Non-interactive key exchange. In K. Kurosawa and G. Hanaoka, editors, *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, volume 7778 of *Lecture Notes in Computer Science*, pages 254–271. Springer, 2013.
63. G. Frey, M. Müller, and H. Rück. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Trans. Information Theory*, 45(5):1717–1719, 1999.
64. J. Fried, P. Gaudry, N. Heninger, and E. Thomé. A kilobit hidden SNFS discrete logarithm computation. In Coron and Nielsen [37], pages 202–231.
65. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, pages 537–554, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

66. S. D. Galbraith. Constructing isogenies between elliptic curves over finite fields. *LMS Journal of Computation and Mathematics*, 2:118–138, 1999.
67. S. D. Galbraith, F. Hess, and N. P. Smart. Extending the GHS Weil descent attack. In *Advances in cryptology — EUROCRYPT 2002 (Amsterdam)*, volume 2332 of *Lecture Notes in Computer Science*, pages 29–44. Springer, Berlin, 2002.
68. S. D. Galbraith, C. Petit, B. Shani, and Y. B. Ti. On the security of supersingular isogeny cryptosystems. In J. H. Cheon and T. Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 63–91, 2016.
69. S. D. Galbraith and B. Smith. Discrete logarithms in generalized Jacobians. *IACR Cryptology ePrint Archive*, 2006:333, 2006.
70. S. D. Galbraith and F. Vercauteren. Computational problems in supersingular elliptic curve isogenies. *To appear in Quantum Information Processing*, 2017.
71. R. P. Gallant, R. J. Lambert, and S. A. Vanstone. Improving the parallelized Pollard lambda search on anomalous binary curves. *Math. Comput.*, 69(232):1699–1705, 2000.
72. P. Gaudry. Fast genus 2 arithmetic based on Theta functions. *J. Mathematical Cryptology*, 1(3):243–265, 2007. `https://eprint.iacr.org/2005/314/`.
73. P. Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *J. Symb. Comput.*, 44(12):1690–1702, 2009.
74. P. Gaudry, F. Hess, and N. P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *J. Cryptology*, 15(1):19–46, 2002.
75. P. Gaudry, E. Thomé, N. Thériault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Math. Comput.*, 76(257):475–492, 2007.
76. L. Grémy and A. Guillevic. DiscreteLogDB, a database of computations of discrete logarithms, 2017. `https://gitlab.inria.fr/dldb/discretelogdb`.
77. A. Guillevic and F. Morain. Discrete logarithms. In El Mrabet and Joye [57], chapter 9.
78. D. Hofheinz, K. Hövelmanns, and E. Kiltz. A modular analysis of the Fujisaki–Okamoto transformation. In Y. Kalai and L. Reyzin, editors, *Theory of Cryptography*, pages 341–371, Cham, 2017. Springer International Publishing.
79. D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In A. Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 553–571. Springer, 2007.
80. D. Jao and L. De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In B.-Y. Yang, editor, *Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34, Berlin, Heidelberg, 2011. Springer Berlin / Heidelberg.
81. D. Jao, S. D. Miller, and R. Venkatesan. Expander graphs based on GRH with an application to elliptic curve cryptography. *Journal of Number Theory*, 129(6):1491–1504, June 2009.
82. T. Kleinjung, C. Diem, A. K. Lenstra, C. Priplata, and C. Stahlke. Computation of a 768-bit prime field discrete logarithm. In Coron and Nielsen [37], pages 185–201.
83. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

84. N. Koblitz. Hyperelliptic cryptosystems. *J. Cryptology*, 1(3):139–150, 1989.
85. N. Koblitz, editor. *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*. Springer, 1996.
86. D. R. Kohel. *Endomorphism rings of elliptic curves over finite fields*. PhD thesis, University of California at Berkley, 1996.
87. D. R. Kohel, K. Lauter, C. Petit, and J.-P. Tignol. On the quaternion $\ell$-isogeny path problem. *LMS Journal of Computation and Mathematics*, 17(A):418–432, 2014.
88. G. Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal of Computing*, 35(1):170–188, 2005.
89. G. Kuperberg. Another Subexponential-time Quantum Algorithm for the Dihedral Hidden Subgroup Problem. In S. Severini and F. Brandao, editors, *8th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2013)*, volume 22 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20–34, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
90. A. Langley, M. Hamburg, and S. Turner. Elliptic curves for security. *RFC*, 7748:1–22, 2016.
91. A. K. Lenstra and H. W. Lenstra, Jr., editors. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer, Berlin, Heidelberg, 1993.
92. A. K. Lenstra and E. R. Verheul. The XTR public key system. In M. Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2000.
93. C. H. Lim and P. J. Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In B. S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, pages 249–263, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.
94. M. Lochter and J. Merkle. Elliptic curve cryptography (ECC) brainpool standard curves and curve generation. *RFC*, 5639:1–27, 2010.
95. M. Marlinspike and T. Perrin. The X3DH key agreement protocol, 2016.
96. E. Martin-Lopez, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'Brien. Experimental realization of Shor's quantum factoring algorithm using qubit recycling. *Nat. Photon.*, 6(11):773–776, 11 2012.
97. U. M. Maurer. Towards the equivalence of breaking the Diffie–Hellman protocol and computing discrete algorithms. In Y. Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 271–281. Springer, 1994.
98. U. M. Maurer and S. Wolf. The relationship between breaking the Diffie–Hellman protocol and computing discrete logarithms. *SIAM J. Comput.*, 28(5):1689–1721, 1999.
99. G. Maze, C. Monico, and J. Rosenthal. Public key cryptography based on semigroup actions. *Adv. in Math. of Comm.*, 1(4):489–507, 2007.
100. A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Information Theory*, 39(5):1639–1646, 1993.

101. J. Mestre. La méthode des graphes. Exemples et applications. In *Proceedings of the international conference on class numbers and fundamental units of algebraic number fields (Katata)*, pages 217–242, 1986.
102. V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1985.
103. D. J. Mireles Morales. An analysis of the infrastructure in real function fields. *IACR Cryptology ePrint Archive*, 2008:299, 2008.
104. P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.
105. V. K. Murty. Abelian varieties and cryptography. In S. Maitra, C. E. V. Madhavan, and R. Venkatesan, editors, *Progress in Cryptology - INDOCRYPT 2005, 6th International Conference on Cryptology in India, Bangalore, India, December 10-12, 2005, Proceedings*, volume 3797 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.
106. A. Muzereau, N. P. Smart, and F. Vercauteren. The equivalence between the DHP and DLP for elliptic curves used in practical applications. *LMS Journal of Computation and Mathematics*, 7:50–72, 2004.
107. National Institute of Standards and Technology (NIST). SP 800-56A recommendations for pair-wise key-establishment schemes using discrete logarithm cryptography.
108. NIST. Post-quantum cryptography standardization.
109. E. Ochoa–Jiménez, F. Rodriguez–Henríquez, and M. Tibouchi. Discrete logarithms. In El Mrabet and Joye [57], chapter 8.
110. C. Peikert. Lattice cryptography for the internet. In M. Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, volume 8772 of *Lecture Notes in Computer Science*, pages 197–219. Springer, 2014.
111. T. Perrin and M. Marlinspike. The double ratchet algorithm, 2016.
112. C. Petit. Faster algorithms for isogeny problems using torsion point images. In Takagi and Peyrin [131], pages 330–353.
113. S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (corresp.). *IEEE Trans. Information Theory*, 24(1):106–110, 1978.
114. J. M. Pollard. Monte Carlo methods for index computation (mod $p$). *Mathematics of Computation*, 32(143):918–924, 1978.
115. O. Regev. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. arXiv:quant-ph/0406151, June 2004.
116. J. Renes, P. Schwabe, B. Smith, and L. Batina. $\mu$Kummer: Efficient hyperelliptic signatures and key exchange on microcontrollers. In B. Gierlichs and A. Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 301–320. Springer, 2016.
117. E. Rescorla. The transport layer security (TLS) protocol version 1.3. *RFC*, 8446:1–160, 2018.
118. D. Robert. *Theta functions and cryptographic applications*. PhD thesis, Université Henri Poincaré - Nancy I, July 2010.

119. M. Roetteler, M. Naehrig, K. M. Svore, and K. E. Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In Takagi and Peyrin [131], pages 241–270.
120. A. Rostovtsev and A. Stolbunov. Public-key cryptosystem based on isogenies. *IACR Cryptology ePrint Archive*, 2006:145, 2006.
121. K. Rubin and A. Silverberg. Torus-based cryptography. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 349–365. Springer, 2003.
122. C.-P. Schnorr. Efficient identification and signatures for smart cards. In Brassard [26], pages 239–252.
123. D. Shanks. Class number, a theory of factorization and genera. *Proc. Symp. PureMath.*, 20:415–440, 1971.
124. P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. IEEE, 1994.
125. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.
126. J. H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1992.
127. N. P. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology*, 12(3):193–196, 1999.
128. B. Smith. Isogenies and the discrete logarithm problem in Jacobians of genus 3 hyperelliptic curves. *J. Cryptology*, 22(4):505–529, 2009.
129. A. Stolbunov. Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. *Adv. Math. Commun.*, 4(2), 2010.
130. A. V. Sutherland. Accelerating the CM method. *LMS J. Comput. Math.*, 15:172–204, 2012.
131. T. Takagi and T. Peyrin, editors. *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*. Springer, 2017.
132. S. Tani. Claw finding algorithms using quantum walk. *Theor. Comput. Sci.*, 410(50):5285–5297, 2009.
133. E. Thormarker. *Post-Quantum Cryptography: Supersingular Isogeny Diffie–Hellman Key Exchange*. PhD thesis, Stockholm University, 2017.
134. D. Urbanik and D. Jao. SoK: The problem landscape of SIDH. In *Proceedings of the 5th ACM on ASIA Public-Key Cryptography Workshop*, APKC '18, pages 53–60, New York, NY, USA, 2018. ACM.
135. W. van Dam, S. Hallgren, and L. Ip. Quantum algorithms for some hidden shift problems. *SIAM Journal on Computing*, 36(3):763–778, 2006.
136. A. van der Poorten and A. Stein, editors. *High Primes and Misdemeanors: Lectures in Honour of the 60th Birthday of Hugh Cowie Williams*, volume 42 of *Fields Institute Communications Series*. American Mathematical Society.
137. J. Vélu. Isogénies entre courbes elliptiques. *C. R. Acad. Sci. Paris Sér. A-B*, 273:A238–A241, 1971.

138. E. Wenger and P. Wolfger. Harder, better, faster, stronger: elliptic curve discrete logarithm computations on FPGAs. *J. Cryptographic Engineering*, 6(4):287–297, 2016.
139. M. J. Wiener and R. J. Zuccherato. Faster attacks on elliptic curve cryptosystems. In S. E. Tavares and H. Meijer, editors, *Selected Areas in Cryptography '98, SAC'98, Kingston, Ontario, Canada, August 17-18, 1998, Proceedings*, volume 1556 of *Lecture Notes in Computer Science*, pages 190–200. Springer, 1998.