Tools for designing protocols based on isogenies

Luca De Feo

Isogeny School 2020

Abstract

Post-quantum isogeny based cryptography starts with key exchange (SIDH, CSIDH), and often ends with it. Even isogeny based signatures have taken several years to develop, and are often highly technical. But from such a rich family of assumptions much more is expected than just key exchange or signatures.

Redesigning from scratch any new primitive is a time-consuming and error-prone task. It is much easier to abstract away the complexity of a mathematical construction into a framework that lowers entry barriers and simplifies protocol design. Think about how discrete logarithm groups simplify thinking about elliptic curve cryptography, or of the myriad applications of pairing groups.

Unfortunately, not all of isogeny based cryptography appears to be amenable to simple and powerful abstractions. These lecture notes are about that part of isogeny based cryptography that is. We first define the frameworks, staying clear of the technical complications, then we present some protocols constructed with them.

These notes are divided in two parts. The first part deals with *isogenous pairing groups*, a combination of pairing based and isogeny based cryptography that leads to interesting *time-delay protocols*. Alas, the use of pairings make these protocols not quantum-safe.

The second part deals with *cryptographic group actions*, i.e., essentially with CSIDH. We formalize the pitfalls that make building upon CSIDH harder than we would like it to be, then we build upon it anyway.

Exercises are scattered along the way. Most of them are standard exercises the reader may already be familiar with. We also give some problems: these may go from little explored research questions to big open questions in the field.

Part I
Isogenous Pairing Groups

It all starts with an equation:

$$e'_N\bigl(\phi(P), Q\bigr) = e_N\bigl(P, \hat\phi(Q)\bigr). \tag{1}$$

Let's dissect it:

- $\phi : E \to E'$ is an isogeny from an elliptic curve E to an elliptic curve E';
- $\hat{\phi} : E' \to E$ is the dual of ϕ;
- N is a positive integer, usually a prime;
- $P \in E[N]$ is an N-torsion point on the *domain* curve;
- $Q \in E'[N]$ is an N-torsion point on the *image* curve;
- e_N and e'_N are pairings of order N on E and E' respectively, usually the Weil pairings of E and E'.

That this equation is satisfied for any choice of ϕ, N, P, Q and for any known elliptic pairing is a remarkable fact, the proof of which is out of the scope of these notes. See [39, § III.8] for the details. Our goal here is to exploit Eq. (1) to construct new cryptographic protocols.

1 Pairings

We take a step back and recall the basic definitions and properties of cryptographic pairings. If you are already familiar with them, you can definitely skip this section.

Definition 1. *A pairing of two groups G_1, G_2 is a bilinear map $e : G_1 \times G_2 \to G_3$, i.e., one such that:*

- $e(g^a, h) = e(g, h^a) = e(g, h)^a$,
- $e(gg', h) = e(g, h)e(g', h)$,
- $e(g, hh') = e(g, h)e(g, h')$,

for all $a \in \mathbb{Z}$, all $g, g' \in G_1$ and all $h, h' \in G_2$.

A pairing is said to be non-degenerate *if:*

- $e(g, h) = 1$ *for all g implies $h = 1$, and*
- $e(g, h) = 1$ *for all h implies $g = 1$.*

A pairing is said to be alternating *if $G_1 = G_2$ and $e(g, g) = 1$ for all g.*

Exercise 1. *Let e be alternating, prove that $e(g, h) = e(h, g)^{-1}$.*

Remark 2. *In the definition above we denoted all three groups G_1, G_2, G_3 multiplicatively, which is the standard convention in cryptography.*

However, in practice, elliptic pairings have groups of elliptic points for G_1 and G_2, and a multiplicative subgroup of a finite field for G_3. Thus, textbooks on elliptic curves tend to denote G_1 and G_2 additively, but G_3 multiplicatively, e.g.: $e(P + Q, R) = e(P, R)e(Q, R)$. I think we can all agree this is extremely confusing to anyone except arithmetic geometers, and we shall thus avoid this devilish notation.

For a pairing to be useful in cryptography, it needs to be easy to compute, but it also needs some hardness assumptions. There is a plethora of different assumptions in the literature, but we shall restrict our attention to only a few:

- The groups G_1, G_2, G_3 are (generalized) discrete logarithm groups, in particular they are usually assumed to have prime order;
- *Pairing inversion* (see below) is hard.

Definition 3 (Pairing inversion problem)**.** *Let $e : G_1 \times G_2 \to G_3$ be a pairing, the* pairing inversion *problem on G_1 (resp. G_2) asks, given $t \in G_3$ and $h \in G_2$ (resp. $g \in G_1$), to find a g (resp. h) such that*

$$e(g, h) = t.$$

Exercise 2. *Assume that G_1, G_2, G_3 are cyclic. Prove that pairing inversion is no harder than discrete logarithm (in which group?).*

Exercise 3. *Let G be a cyclic group, and let $e : G \times G \to G_3$ be a non-degenerate pairing, prove that DDH is easy in G.*

2 The Weil pairing

Elliptic curve subgroups come equipped with a natural pairing. Let E/k be an elliptic curve defined over a field k, and let N be a positive integer prime to the characteristic of k. We know that the torsion subgroup $E[N]$ has rank two, i.e., $E[N] \simeq (\mathbb{Z}/N\mathbb{Z})^2$. The Weil pairing of order N is a non-degenerate alternating pairing $e_N : E[N] \times E[N] \to \mu_N$, where $\mu_N \subset \bar{k}$ is the subgroup of N-th roots of unity of (the algebraic closure of) k.

Exercise 4 (warning: multiplicative notation!)**.** *Let $\langle P, Q\rangle$ be a basis of $E[N]$, show that*

$$e_N(P^aQ^b, P^cQ^d) = e_N(P, Q)^{\det\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)},$$

which is equal to 1 if and only if the determinant is 0 (modulo N).

It is not important here to know how the Weil pairing is computed (see, e.g., [25]). Suffice to say that there is an efficient algorithm (with running time polynomial in $\log(N)$). Other pairings are also defined for elliptic curves, however they are all related to the Weil pairing, and they will not make a difference for our purposes, thus we will ignore them. For a review of known elliptic pairings, addressed to non-specialists, see [26].

Eq. (1) is the main (the only) result we care about. Let's hammer it home.

Theorem 4. *Let E, E' be elliptic curves, let $\phi : E \to E'$ be an isogeny, $\hat{\phi} : E' \to E$ its dual, let N be a positive integer. For any $P \in E[N]$ and $Q \in E'[N]$*

$$e'_N\big(\phi(P), Q\big) = e_N\big(P, \hat{\phi}(Q)\big). \tag{1}$$

Remark 5. *Eq.* (1) *is a generalization of the bilinearity of the Weil pairing. Indeed, for any integer a, the multiplication-by-a map $[a] : E \to E$ is an endomorphism (thus, an isogeny) of E, and its dual is $[a]$ itself, thus*

$$e_N([a]P, Q) = e_N(P, [a]Q).$$

Since we have opted for the multiplicative notation P^a, rather than $[a]P$, it makes sense to also treat isogenies like exponents. At the cost of facing backlash from the whole community, we will from now on rewrite Eq. (1) *as*

$$e'_N\big(P^{\phi}, Q\big) = e_N\big(P, Q^{\hat{\phi}}\big). \tag{1'}$$

For clarity, we will always use Greek letters for isogenies and Latin letters for integer exponents.

Exercise 5. *Let $\phi : E \to E'$ be an isogeny of degree d. Prove that, for any N, P, Q*

$$e'_N\big(P^{\phi}, Q^{\phi}\big) = e_N(P, Q)^d.$$

3 Pairing-friendly curves

The Weil pairing is defined for any curve and any $N > 0$, however constructing curves for which it has the desired cryptographic properties is an art. Concretely, we want to satisfy a few constraints:

- The curve must be defined over a finite field $\mathbb{F}_p$;
- N must be large and prime;
- The torsion points $E[N]$ must be defined over an extension of $\mathbb{F}_p$ of small degree.

The last constraint is where all the difficulty lies. There exist a few different constructions to achieve it, but we will only care about one.

Theorem 6. *Let $p > 3$ be a prime, let E be a supersingular curve:*

- *defined over $\mathbb{F}_p$, or*
- *defined over $\mathbb{F}_{p^2}$ such that $\#E(\mathbb{F}_{p^2}) = (p + 1)^2$,*

let $N|(p + 1)$. Then $E[N] \subset E(\mathbb{F}_{p^2})$.

4 Quantum annoying

Eq. (1) is obviously central to the theory of elliptic curves and isogenies, however it is not immediately clear of what use it could be in cryptography. Indeed, while Eq. (1) generalizes bilinearity, it is also more constrained, thus we expect to be able to build strictly less protocols than with pairings alone.

However, isogeny problems tend to be quantum-resistant, while pairing problems are certainly not (see Exercise 2). Mixing them cannot possibly lead to a quantum-resistant protocol, however, it can at least offer some *partial* security guarantees against quantum attackers. The first to suggest this were Koshiba and Takashima [28, 29], who introduced the name "Isogenous Paring Groups" (IPG), and who applied them to identity-based and attribute-based encryption with the goal of offering security against an adversary with limited access to quantum resources.

More recently, the concept of *quantum annoying* has been formalized for password-based protocols [24]. Intuitively, a password-based protocol (e.g., a Password Authenticated Key Exchange, or PAKE) is "quantum annoying" if the best strategy for an adversary requires computing a new discrete logarithm for each password guess. Formalizing quantum annoying-ness is still work in progress. Rather than going 100% formal, here we will just review some very basic primitives, which are believed (or is it just me?) to be quantum annoying to some extent.

4.1 PRF

A cryptographic *pseudorandom function* (PRF) is, roughly speaking, function $f : X \to Y$ that is computationally indistinguishable from a truly random function. We are interested in a weaker property, called *unpredictability.*

Definition 7 (wUF)**.** *Let K, X and Y be finite sets. A* weakly unpredictable function family *(wUF) is a family of efficiently computable* keyed *functions $\{f_k : X \to Y \mid k \in K\}$ such that for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ has negligible advantage at the following game:*

1. *A random $k \in K$ is drawn uniformly;*

2. *$\mathcal{A}$ can query a randomized oracle O_k that takes no input and outputs uniformly random pairs $(x, f_k(x))$;*

3. *$\mathcal{A}$ requests a challenge and receives a uniformly random $x \in X$, from this moment on $\mathcal{A}$ has no more access to O_k;*

4. *$\mathcal{A}$ wins if it outputs $f_k(x)$.*

Exercise 6. *Let G be a group. The family $\{f_k : g \mapsto g^k \mid 1 \leq k \leq \#G\}$ is a wUF if and only if CDH is hard in G.*

Analogous statements are true for isogenies, however some care must be applied in defining the family.

Exercise 7. *Let E and E' be isogenous curves, let N be an integer. Let $\phi : E \to E'$ be an isogeny of degree coprime to N, let f_ϕ be the restriction of ϕ to $E[N]$. Propose an algorithm to win the wUF game (i.e., predict the output of f_ϕ) using a discrete logarithm solver.*

Although CDH is possibly an easier problem than DLP, note that the only known way to solve CDH is to compute one discrete logarithm. Thus, if for quantum annoying-ness we are interested in counting the number of calls to a DL oracle, the reduction in Exercise 6 and that in 8 cost the same.[1]

Exercise 8. *Let E and E' be isogenous curves, let N be an integer. Describe one or more families of isogenies $f : E \to E'$ such that their restriction to $E[N]$ forms a wUF.*

In terms of quantum annoying-ness, there is a huge difference between the family in Exercise 6 and those you may have defined in 8. In the first case, one discrete logarithm is enough to recover k, then $\mathcal{A}$ can compute as many values $f_k(x)$ as they like. In the second case, it is not clear how to do it without computing a discrete logarithm for each new challenge x.

Problem 9. *Following [24], formalize quantum annoying-ness for wUFs, and find families of isogenies that possess the property.*

4.2 VRF

A *verifiable random function* (VRF) is, roughly speaking, a wUF that is not a PRF.

Definition 8 (VRF)**.** *A* verifiable random function family *(VRF) is a wUF $\{f_k : X \to Y \mid k \in K\}$ such that there exists algorithms:*

- *Keygen, taking k as input and outputting a public key pk;*
- *Prove, taking as input $x, k, f_k(x)$ and outputting a proof π;*
- *Verify, taking as input x, y, pk, π and outputting 1 if and only if $y = f_k(x)$.*

The following example is none else than the building block of the celebrated BLS signature scheme [10].

Exercise 10. *$e : G_1 \times G_2 \to G_3$ be a non-degenerate pairing on CDH groups, show that the family $\{f_k : G_1 \to G_1 : g \mapsto g^k \mid 1 \le k \le \#G_1\}$ is a VRF.*

The construction above is easily extended to isogenous pairing groups. In [18] it is claimed, without a security reduction, that such extension is quantum annoying.

Problem 11. *Formalize the notion of quantum annoying-ness for VRFs, and find families of isogenies that possess the property.*

[1]In the generic group model, CDH and DLP are known to be roughly equivalent [38, 31].

4.3 (V)OPRF

An *oblivious PRF* (OPRF) is a protocol that lets two parties jointly compute a PRF without revealing secret information to each other.

Definition 9 (OPRF)**.** *A PRF $\{f_k : X \to Y \mid k \in K\}$ can be computed* obliviously *if there exist a protocol between two parties $\mathcal{S}$ and $\mathcal{C}$ such that:*

- *at the beginning of the protocol, $\mathcal{S}$ knows $k \in K$ and $\mathcal{C}$ knows $x \in X$;*
- *at the end of the protocol $\mathcal{C}$ learns $f_k(x)$ and $\mathcal{S}$ learns nothing.*

No information is revealed to $\mathcal{C}$ and $\mathcal{S}$ other than that learned in the ideal protocol.

Let G be a group, let H_1 be a hash function with range G, and let H_2 be another hash function. The family $\{f_k : x \mapsto H_2(x, H_1(x)^k) \mid 1 \leq k \leq \#G\}$ can be computed obliviously thanks to a protocol introduced in [27]:

1. On input x, $\mathcal{C}$ selects a random r and sends $H_1(x)^r$ to $\mathcal{S}$;
2. Upon receiving $a = H_1(x)^r$ from $\mathcal{C}$, $\mathcal{S}$ sends back a^k;
3. Upon receiving $b = H_1(x)^{rk}$ from $\mathcal{S}$, $\mathcal{C}$ computes $f_k(x) = H_2(x, b^{1/r})$.

A *verifiable OPRF* (VOPRF) is one with algorithms Keygen, Prove and Verify, that let $\mathcal{C}$ verify that $\mathcal{S}$ behaves honestly (i.e., does not lead $\mathcal{C}$ to compute $f_{k'}(x)$ for some $k' \neq k$). Note that a VOPRF is not necessarily a VRF, as the output $f_k(x)$ needs not be publicly verifiable.

Problem 12. *Define quantum annoying-ness for (V)OPRF, then propose an instantiation using the IPG framework.*

5 Isogeny walks and sequential computation

Orthogonally to quantum annoying-ness, rooted deeper in the theory of isogeny graphs, IPG have recently attracted more interest thanks to the "sequentiality" properties of isogeny computations.

Isogeny computations are notoriously slow, but here we're talking about another level of slowness. The goal of *time-delay cryptography* is to design protocols where one party must compute for a given amount of *time*, and plausibly no less. Mathematically defining time is not really feasible, but a good practical approximation is to define a *sequential computation*: a sequence of small elementary steps that must be executed in a set order, and such that the final result cannot, conjecturally, be computed in any other faster way. This model of secure computation is radically different from the usual complexity-theoretic model of cryptography, where, for example, the solution space of an NP problem can be brute-forced in parallel, rendering moot any attempt at lower-bounding *time*.

An example of sequential computation is iterated hashing. Let H be a hash function, and define the function

$$f(x) := H^n(x) = H(H(\cdots(H(x)))).$$

Assuming the output of H is in some sense "unpredictable", we can affirm that computing $f(x)$ will take no less time than evaluating n times H. Then, to lower bound *wall time* we can focus on the best possible hardware implementation of H, not unlike what is currently done with the *proof of work* of the most popular cryptocurrencies.

Another example of computation that is conjectured to be sequential is *repeated squaring* in groups of unknown order. Let G be a group, define the function

$$f(x) := x^{2^n}.$$

If N is the order of G, then $x^{2^n} = x^{2^n \bmod N}$, and thus the value of $f(x)$ could be computed with only $O(\log(N))$ group operations. However when the order N is supposed unknown, no better algorithm is known to compute $f(x)$ other than squaring n times. This function, possessing more structure than iterated hashing, has been used to define efficient *verifiable delay functions* (VDF) [34, 41], that we shall define in the next section. Note that groups of unknown order (e.g., RSA groups) are an intrinsically classical cryptographic construct, as Shor's quantum algorithm can compute the order of any group.

Here we are interested in the sequential properties of *evaluating isogeny walks*. Let ϕ be an isogeny of degree, say, 2^n. Then, ϕ factors as a walk

$$E_0 \xrightarrow{\phi_1} E_1 \xrightarrow{\phi_2} \cdots \xrightarrow{\phi_n} E_n$$

of isogenies ϕ_i of degree 2. If we fix an integer N, we may hope that evaluating the restriction $\phi : E_0[N] \to E_n[N]$ is a sequential computation requiring no less than n evaluations of isogenies of degree 2. Since isogenies of degree 2 can be evaluated by a small fixed amount of finite field operations, we may focus on designing the best possible hardware realization of 2-isogeny evaluation, and then use n times the latency of that hardware as a lower bound.

However, like for repeated squaring, some known structure helps simplify computations. It is indeed possible that for a given isogeny walk $\phi : E_0 \to E_n$, there exist shorter related isogeny walks $\psi : E_0 \to E_n$ that let us compute x^ϕ in less time. The technical details of when such *shortcuts* exist, and how to computed them are presented in [18]. Here we just apply the following rules of thumb, that have been developed in the previous courses on class group computations and the KLPT algorithm:

- Cycles in an isogeny graph correspond to endomorphisms, walks correspond to ideals in $\mathrm{End}(E)$.
- Given enough cycles, we can compute $\mathrm{End}(E)$; given $\mathrm{End}(E)$, we can compute cycles.

- Given $\mathrm{End}(E)$ and a walk $E \to E'$, we can compute $\mathrm{End}(E')$; given $\mathrm{End}(E)$ and $\mathrm{End}(E')$, we can compute a walk $E \to E'$.
- Given $\mathrm{End}(E)$ and a walk $\phi : E \to E'$, we can compute the corresponding ideal $I \subset \mathrm{End}(E)$; given $\mathrm{End}(E)$ and an ideal $I \subset \mathrm{End}(E)$, we can compute the corresponding walk.
- Given $\mathrm{End}(E)$ and an ideal $I \subset \mathrm{End}(E)$, we can compute ideals $J \subset \mathrm{End}(E)$ in the same class as I.

All these rules hold (provided the appropriate bounds on isogeny degrees and ideal norms) regardless of whether $\mathrm{End}(E)$ is commutative or not.

The moral teaching is that if we know $\mathrm{End}(E_0)$, and if we have a walk $\phi : E_0 \to E_n$, we can probably compute, if it exists, a shorter walk $\psi : E_0 \to E_n$. For the kind of walks ϕ we are interested in, several millions of steps long, much shorter walks are always expected to exist.

The way around this problem is, like in the repeated squaring case, to assume that $\mathrm{End}(E_i)$ is unknown. With the current knowledge, this appears to be impossible for ordinary curves[2], but for supersingular curves the difficulty of computing endomorphism rings is a standard assumption, underpinning all of isogeny based cryptography.

Finding *shortcuts* is not the only way to beat sequentiality. The following exercise shows that sequentiality of isogeny walk evaluations cannot hold in a quantum world.

Exercise 13. *Let $\phi : E_0 \to E_n$ be an isogeny walk, let $N \ll 2^n$ be an integer. Given $g \in E_0[N]$ and g^ϕ, and given access to a DLP oracle, explain how to compute h^ϕ for a random $h \in E_0[N]$ in only $O(\log(N))$ steps.*

Despite the "easy patch" to prevent the computation of *shortcuts*, assuming $\mathrm{End}(E)$ is unknown is easier said than done. We conclude with one of the most fascinating open problems in isogeny based cryptography, and we refer to [18, 12] for more details.

Problem 14. *Find an algorithm that samples a random supersingular curve E, and such that computing $\mathrm{End}(E)$ is difficult even when given access to the internal state of the algorithm.*

6 Delay protocols

Sequentiality alone is not very useful. In time-delay cryptography we seek to build protocols where one party needs to go through the slow sequential computation, while the other parties only have efficient computations. Such protocols roughly fall within two categories:

[2] More precisely, we do not know how to construct ordinary *pairing friendly* curves such that it is hard to compute their endomorphism ring. Pairings are not necessary for sequentiality, but will be needed in the next section.

- Protocols akin to signatures, where the goal is to *efficiently verify* that one party honestly performed the slow sequential computation;
- Protocols akin to encryption, where the goal is to *efficiently encrypt* a message that will take a long sequential computation to decrypt.

To the first category belong *proofs of work* (PoW) and *verifiable delay functions* (VDF) [7]. To the second belong *time-lock puzzles* (TLP) [35] and *delay encryption* (DE) [12].

A crucial difference between the protocols presented in Section 4 and delay protocols is that the latter have no secrets. This slightly simplifies security definitions, but brings in the burden of unknown structure assumptions mentioned in the previous section.

A VDF is, roughly speaking, a VRF with slow evaluation. It is a special instance of PoW, where an input uniquely determines the output.

Definition 10 (VDF)**.** *A* verifiable delay function *(VDF) consists of three algorithms:*

$\mathsf{Setup}(\lambda, T) \to (\mathsf{ek}, \mathsf{vk})$**.** *is a procedure that takes a security parameter λ, a delay parameter T, and outputs public parameters consisting of an evaluation key* ek *and a verification key* vk.

$\mathsf{Eval}(\mathsf{ek}, s) \to (a, \pi)$**.** *is a procedure to evaluate the function on input s. It produces the output a from s, and a (possibly empty) proof π. This procedure is meant to be infeasible in time less than T.*

$\mathsf{Verify}(\mathsf{vk}, s, a, \pi) \to \{\mathsf{true}, \mathsf{false}\}$**.** *is a procedure to verify that a is indeed the correct output for s, with the help of the proof π.*

A VDF shall satisfy three security properties: *Correcteness*, stating that a honest evaluator always passes verification, *Soundness*, stating that a lying evaluator never passes verification, and *Sequentiality*, stating that it is impossible to correctly evaluate the VDF in time less than $T - o(T)$, even when using $\mathrm{poly}(T)$ parallel processors.

Exercise 15. *Drawing inspiration from the VRF of Exercise 10, design an isogeny based VDF.*

Delay encryption is related to identity based encryption [8]. Rather then encrypting to an identity, parties encrypt to an ephemeral *session.* Once a session is started, participants can start *extracting* the session decryption key, a slow sequential operation. After extraction is completed, anyone in possession of the session key can decrypt all messages encrypted to the session.

Definition 11 (DE)**.** *A* delay encryption *scheme consists of four algorithms:*

$\mathsf{Setup}(\lambda, T) \to (\mathsf{ek}, \mathsf{pk})$**.** *Takes a* security parameter λ*, a* delay parameter T*, and produces public parameters consisting of an* extraction key ek *and an* encryption key pk. Setup *must run in time* $\mathrm{poly}(\lambda, T)$*; the encryption key* pk *must have size* $\mathrm{poly}(\lambda)$*, but the evaluation key* ek *is allowed to have size* $\mathrm{poly}(\lambda, T)$.

$\mathsf{Extract}(\mathsf{ek}, \mathsf{id}) \to \mathsf{idk}$**.** *Takes the extraction key* ek *and a* session identifier $\mathsf{id} \in \{0,1\}^*$*, and outputs a* session key idk. $\mathsf{Extract}$ *is expected to run in time* exactly T.

$\mathsf{Encaps}(\mathsf{pk}, \mathsf{id}) \to (c, k)$**.** *Takes the encryption key* pk *and a* session identifier $\mathsf{id} \in \{0,1\}^*$*, and outputs a* ciphertext $c \in \mathcal{C}$ *and a* key $k \in \mathcal{K}$. Encaps *must run in time* $\mathrm{poly}(\lambda)$.

$\mathsf{Decaps}(\mathsf{pk}, \mathsf{id}, \mathsf{idk}, c) \to k$**.** *Takes the encryption key* pk*, a* session identifier id*, a* session key idk*, a ciphertext* $c \in \mathcal{C}$*, and outputs a key* $k \in \mathcal{K}$. Decaps *must run in time* $\mathrm{poly}(\lambda)$.

When Encaps and Decaps are combined with a symmetric encryption scheme keyed by k, they become the encryption and decryption routines of a hybrid encryption scheme.

A Delay Encryption scheme is correct if for any $(\mathsf{ek}, \mathsf{pk}) = \mathsf{Setup}(\lambda, T)$ and any id

$$\mathsf{idk} = \mathsf{Extract}(\mathsf{ek}, \mathsf{id}) \;\wedge\; (c, k) = \mathsf{Encaps}(\mathsf{pk}, \mathsf{id}) \;\Rightarrow\; \mathsf{Decaps}(\mathsf{pk}, \mathsf{id}, \mathsf{idk}, c) = k.$$

The security of Delay Encryption is defined similarly to that of public key encryption schemes, and in particular of identity-based ones; however one additional property is required of $\mathsf{Extract}$: that for a randomly selected identifier id, the probability that any algorithm outputs idk in time less than T is negligible.

Exercise 16. *(hard?) Define a DE scheme using the IPG framework.*

I am unsure how hard this exercise is. If you are stuck, check out [12, § 3]. One notable fact about DE is that this isogeny based construction is the *only known instantiation* of DE.

Part II
Cryptographic Group Actions

Also known as *hard homogeneous spaces* [15], cryptographic group actions are a generalization of discrete logarithm groups that potentially offer resistance to quantum attacks. Currently, the most popular post-quantum group action is undoubtedly CSIDH [13], but the concept goes back to Couveignes' and Rostovtsev and Stolbunov's key exchange scheme based on the group action of complex multiplication on ordinary elliptic curves [15, 36]. Even earlier, group actions were studied in the context of cryptography by Brassard and Yung [11].

Like discrete logarithm groups, group actions offer a simple but powerful abstraction, from which we may derive many different primitives. Like with any abstraction, though, there is a risk of loosing sight of the actual mathematical object, and defining a protocol that cannot be instantiated or, worse, is insecure.

In this part, we describe cryptographic group actions using the formalism of [1]. This framework, more detailed than the early attempt of [15], closely models the actual properties of CSIDH and its derivatives.

7 Effective group actions

Definition 12 (Group Action)**.** *Let G be a group with identity element 1, and let X be a set. A map $\star : G \times X \to X$ is called an* action of G on X *if:*

- *$1 \star x = x$ for any $x \in X$, and*
- *$(gh) \star x = g \star (h \star x)$ for any $g, h \in G$ and any $x \in X$.*

A group action $(G, X, \star)$ is called transitive *if for every $x_1, x_2 \in X$, there exists $g \in G$ such that $x_2 = g \star x_1$. It is called* free *if for every $g \in G$, whenever there is some $x \in X$ such that $x = g \star x$, then $g = 1$. A* regular *group action is one that is transitive and free.*

Exercise 17. *Let $(G, X, \star)$ be a group action. Prove that for any $g \in G$ the map $\pi_g : x \mapsto g \star x$ is a permutation of X.*

Exercise 18. *Let $(G, X, \star)$ be a regular group action. Prove that for any $x \in X$ the map $f_x : g \mapsto g \star x$ defines a bijection between G and X.*

In particular, for a regular group action, if G (or X) is finite then we must have $\#G = \#X$. One informal way to think of X in this case is like a copy of G where we forget which element is the group identity. From this point on, whenever we mention a group action this will be implicitly assumed to be **regular, finite and abelian**.

For a group action to be useful in cryptography, some operations must be easily computable.

Definition 13 (Effective group action — EGA)**.** *A group action $(G, X, \star)$ is* effective *if the following properties are satisfied:*

1. *The group G is finite and there exist efficient algorithms for:*
   (a) Membership testing*, i.e., to decide if a given bit string represents a valid group element in G.*
   (b) Equality testing*, i.e., to decide if two bit strings represent the same group element in G.*
   (c) Sampling*, i.e., to sample an element g from a distribution on G statistically close to uniform.*
   (d) Operation*, i.e., to compute gh for any $g, h \in G$.*
   (e) Inversion*, i.e., to compute g^{-1} for any $g \in G$.*
2. *The set X is finite and there exist efficient algorithms for:*
   (a) Membership testing*, i.e., to decide if a bit string represents a valid set element.*
   (b) Unique representation*, i.e., given any arbitrary set element $x \in X$, compute a string $\hat{x}$ that canonically represents x.*

3. *There exists a distinguished element $x_0 \in X$, called the* origin*, such that its bit-string representation is known.*

4. *There exists an efficient algorithm that given (some bit-string representations of) any $g \in G$ and any $x \in X$, outputs $g \star x$.*

A cryptographic group action also needs to have some hard problems. Recall that a family of efficiently computable functions $f_k : X \to Y$ is *one way* if computing f_k^{-1} is hard.

Definition 14. *Let $(G, X, \star)$ be an EGA. Define the functions*

$$f_x : G \to X, \qquad \pi_g : X \to X,$$
$$g \mapsto g \star x, \qquad x \mapsto g \star x.$$

The group action is said to be:

1. One-way *if the family of functions f_x is one-way.*

2. Weakly unpredictable *if the family of permutations π_g is weakly unpredictable, i.e., if given a list of random pairs $(x, \pi_g(x))$ it is hard to guess $\pi_g(x^*)$ for a random x^* not in the list.*

3. Weakly pseudorandom *if the family of permutations π_g is weakly pseudorandom, i.e., if it is hard to distinguish between a list of random pairs $(x, \pi_g(x))$ and one of random pairs $(x, \pi(x))$, where π is a uniformly drawn permutation of X.*

Exercise 19. *Let G be an effective group of order p. Define an effective group action on G so that one-wayness is equivalent to DLP. What are unpredictability and pseudorandomness equivalent to?*

Exercise 20. *Instantiate key exchange and public key encryption from an EGA. What assumption can the security of these primitives be based on?*

8 Restricted effective group actions

It may come as a surprise that no *strictu sensu* post-quantum EGA is currently known. The group action of complex multiplication used in CSIDH and related protocols comes close to a pq-EGA, however it fails to satisfy the last axiom: efficiently evaluating $g \star x$ for *any* $g \in G$.

For this reason we introduce a restricted version of EGA, which closely models the algorithmic properties of isogeny group actions. The hardness assumptions will stay unchanged.

Definition 15 (Restricted effective group action — REGA)**.** *Let $(G, X, \star)$ be a group action and let $\vec{g} = (g_1, \dots, g_n)$ be a (not necessarily minimal) generating set for G. The action is said to be* $\vec{g}$-restricted effective*, if the following properties are satisfied:*

1. *G is finite and $n = \mathrm{polylog}(\#G)$.*

2. *The set X is finite and there exist efficient algorithms for:* Membership testing*, i.e., to decide if a bit string represents a valid set element; and* Unique representation*, i.e., to compute a string $\hat{x}$ that canonically represents any given set element $x \in X$.*

3. *There exists a distinguished element $x_0 \in X$, called the* origin*, such that its bit-string representation is known.*

4. *There exists an efficient algorithm that given any $1 \le i \le n$ and any bit string representation of $x \in X$, outputs $g_i \star x$ and $g_i^{-1} \star x$.*

In the last axiom, we see how REGAs differ from EGAs: in this case, there is only a short list of group elements g_i for which we can efficiently compute $g_i \star x$. This corresponds well to the case of complex multiplication, where we can only efficiently compute the action of low norm ideals (see lecture notes of previous weeks).

Despite this limitation, we can still evaluate in polynomial time the action of exponentially many elements of G. Indeed, the action of any polynomially sized linear combination $\prod_{i=1}^{n} g_i^{e_i}$ can be evaluated in $\sum_{i=1}^{n} |e_i|$ steps by successively computing the action of each component. For example, restricting exponents to a box $e_i \in [-r, r]$, the action of as many as $(2r+1)^n$ distinct elements of G can be computed. This is precisely how the group action is evaluated in CSIDH.

Remark 16. *Abstractly, a REGA can be viewed as a group action of $\mathbb{Z}^n$ on X, given by*

$$(e_1, \dots, e_n) \star x := \left(\prod g_i^{e_i}\right) \star x,$$

where the cost of evaluating the action depends on the norm of the vectors in $\mathbb{Z}^n$.

The action is not free, however there exists a subgroup $\Lambda \subset \mathbb{Z}^n$, called the relation lattice *of $(g_1, \dots, g_n)$, such that $\vec{e} \star x = x$ if and only if $\vec{e} \in \Lambda$. Then the induced action by $\mathbb{Z}^n/\Lambda$ is regular.*

Exercise 21. *Recast the key exchange and public key encryption protocols of Exercice 20 in the context of REGAs. Is security affected?*

Although we explained how to efficiently evaluate the action of exponentially many elements of G, possibly all of them, we still do not have an EGA. Indeed the representation of elements of G as products $\prod_{i=1}^{n} g_i^{e_i}$ is not unique, and not all such representations can be efficiently evaluated: as soon as the vector of exponents $\vec{e}$ has super-polynomial norm, we fail to satisfy the axioms of EGAs.

Nevertheless, a REGA can be turned into a *pretty good* approximation of an EGA[3] through a pre-computation of the relation lattice. This is the main idea behind the CSI-FiSh signature scheme [6]. The details of the construction

[3]By "pretty good" we mean that this is still not an EGA in the asymptotic sense of complexity theory, but may behave like one in practice.

are out of the scope of these notes, but have been discussed in Ward Beullens' lecture notes [3]. Note that, without this pre-computation, signatures purely based on REGAs are much less efficient [17, 21].

9 Oblivious transfer

Oblivious transfer (OT) is possibly the simplest "advanced" primitive one may design from (R)EGA. It is one of foundational primitives of secure multi-party computation, and a building block for several advanced protocols. There are many variants of OT, but they all aim at realizing roughly the same functionality: a *sender* $\mathcal{S}$ possesses two *messages* m_0, m_1, and a *receiver* $\mathcal{R}$ wishes to learn only one of them, without the sender knowing which.

Several protocols for OT are known from discrete logarithm group, and some of them may be adapted to the (R)EGA setting. One of the simplest such protocols, in the authors' own words, is the Chou–Orlandi scheme [14]. It is a 3-message protocol inspired by the DH key exchange. Given a group $G = \langle g \rangle$ of order p, a hash function H and an encryption scheme (E_k, D_k) (where $E_k(m)$ means "encrypt m under key k" and $D_k(c)$ means "decrypt c"), it proceeds as follows:

1. $\mathcal{S}$ generates a secret $1 \leq s \leq p$, sends $A = g^s$ to $\mathcal{R}$.
2. Let $\beta \in \{0, 1\}$ be a bit indicating the message m_β that $\mathcal{R}$ wishes to learn; $\mathcal{R}$ generates a secret $1 \leq r \leq p$ and sends $B = A^\beta g^r$ to $\mathcal{S}$.
3. $\mathcal{S}$ derives two keys:
   - $k_0 = H(B^s)$, and
   - $k_1 = H((B/A)^s)$;

   it sends $c_0 := E_{k_0}(m_0)$ and $c_1 := E_{k_1}(m_1)$ to $\mathcal{R}$.
4. $\mathcal{R}$ computes $k_\beta = H(A^r)$ and recovers $m_\beta = D_{k_\beta}(c_\beta)$.

Exercise 22. *Prove that the Chou–Orlandi scheme is correct, i.e., that* $\mathcal{R}$ *learns* m_i *at the end of the protocol.*

As described, this OT protocol is only secure against semi-honest adversaries, i.e., *curious* adversaries who follow the protocol, but want to learn information they are not supposed to learn (β in the case of $\mathcal{S}$, or $m_{1-\beta}$ in the case of $\mathcal{R}$).

Exercise 23. *Show that the Chou–Orlandi protocol is statistically secure against curious* $\mathcal{S}$*. What security assumptions are needed to prove security against curious* $\mathcal{R}$*?*

Exercise 24. *Propose a modification of the Chou–Orlandi protocol that works for (R)EGA.* Hint: start by modifying the protocol to use only exponentiation and no multiplication (unlike in steps 2 and 3).

If you are stuck with the last exercise, check out [30, Fig. 2]. As it turns out, this protocol is not secure in the CSIDH setting without using a trusted setup (i.e., a trusted third party generates the system parameters and does not try to learn any secrets afterwards). This is because the CSIDH group action enjoys one special property that is not captured by the axioms of (R)EGA.

Theorem 17 (A *twisted* symmetry in CSIDH)**.** *In the CSIDH group action, there exists a special element $x_0 \in X$ such that given only $g \star x_0$ the element $g^{-1} \star x_0$ can be computed efficiently.*

Exercise 25. *Let $x \in X$ be the origin element (playing the same role as the generator g) in the modified Chou–Orlandi protocol. Explain how $\mathcal{R}$ can cheat if it knows t such that $x = t \star x_0$, where x_0 is the special element above.*

As far as we know, this is the only algebraic property that sets CSIDH apart from a generic REGA. Remarkably, this property can be used to reduce the number of messages in the modified Chou–Orlandi protocol from 3 to 2. See [30, Fig. 2].

Other OT protocols based on isogenies have been proposed in [2, 40, 20, 1].

10 OPRF

We saw that the function $f_k : g \mapsto g^k$ is a wUF if CDH is hard, and it is in fact also weakly pseudorandom if DDH is hard. However f_k is only a wPRF, because it is a group homomorphism: knowing $(x, f_k(x))$, and being allowed to query f_k on x^a, it is easy to distinguish f_k from a random function by checking that $f(x^a) = f(x)^a$.

To break the homomorphism, we can introduce a hash function as we did in Section 4.3: the function

$$f_k : x \mapsto H_2(x, H_1(x)^k),$$

where H_1 is a hash function with range G, is a PRF, assuming DDH is hard and H_1 and H_2 are modeled as random oracles. There is nothing spectacular about a PRF in the ROM, however what's more interesting is that this PRF supports the oblivious evaluation protocol we described in Section 4.3.

It is tempting to try and replicate the construction with a (R)EGA. We know by hypothesis that $\pi_g : x \mapsto g \star x$ is a wPRF. To define a PRF, the first step would be to define a hash function $H_1 : \{0,1\}^* \to X$, and rather one that behaves like a random oracle. The existence of such a function is not postulated by (R)EGA, so it has to be seen whether a specific instantiation supports it.[4] We are out of luck, here: finding such a hash function is a major open problem for isogeny based cryptography, that is essentially equivalent to Problem 14.

[4] The existence of a random oracle with range in a DL group is not supported by the axioms of DL groups either, but it turns out that for most DL groups (e.g., elliptic curves) such functions exist.

A celebrated way to construct PRFs from the DDH assumption alone is due to Naor and Reingold [33]. If $G = \langle g \rangle$ is a group of order p, the key space is defined as $(\mathbb{Z}/p\mathbb{Z})^{n+1}$, and the input space as $\{0,1\}^n$, for some security parameter n. The function family is then defined as

$$f_{k_0,k_1,\dots,k_n} : (b_1,\dots,b_n) \mapsto g^{k_0 \prod_{i=1}^n k_i^{b_i}}.$$

Exercise 26. *Give an analogue of the Naor–Reingold PRF based on EGA. Do you notice any difference if instead we based it on a REGA?*

If in the previous weeks you were fascinated by the isogeny graphs and complex multiplication, the following exercise should appeal to you.

Exercise 27. *Explain the Naor–Reingold PRF in terms of walks in a Cayley graph. Use this explanation to justify pseudorandomness.*

Variants of the Naor–Reingold PRF based on group actions were simultaneously introduced in [1, 9, 32]. It was [9] that introduced an oblivious evaluation protocol for it, using OT as a subroutine. The protocol is amazingly simple. Let $\mathcal{S}$ be the server holding the key $(k_0,\dots,k_n)$, and let $\mathcal{C}$ be the client wanting to evaluate f_k at $(b_1,\dots,b_n)$.

1. $\mathcal{S}$ chooses a random $r_i \in G$ for each $1 \le i \le n$.
2. For each i, client and server engage in an OT protocol to transfer the value $m_i = k_i^{b_i} r_i$ to $\mathcal{C}$.
3. $\mathcal{S}$ computes $x' = (k_0 \prod_{i=1}^n r_i^{-1}) \star x_0$ and sends it to $\mathcal{C}$.
4. $\mathcal{C}$ computes $(\prod_{i=1}^n m_i) \star s'$ to obtain the output value.

Exercise 28. *Prove that the OPRF protocol above is correct. Sketch the proof of security.*

11 Secret sharing "in the exponents"

Secret sharing is a technique to decompose a secret s into n *shares* s_i such that the value s can only be recovered by assembling together some of the shares. The simplest such scheme is *linear secret sharing*, where s is simply decomposed as

$$s = s_1 + s_2 + \cdots + s_n$$

for s_i uniformly sampled in some additive group (e.g., $\mathbb{Z}_2^\lambda$, or $\mathbb{Z}/p\mathbb{Z}$). It is clear that all shares s_i are necessary to recover s.

A more advanced scheme is *Shamir's threshold secret sharing* [37], which uses polynomial interpolation in a finite field to break s into n shares, such that any $k \le n$ of them can be used to reconstruct s. The idea is to sample a random polynomial $f(X)$ of degree $k-1$ with coefficients in $\mathbb{F}_p$, and to define $s = f(0)$ and $s_i = f(i)$.

Exercise 29. *Give an algorithm to compute s from k shares s_i "without recomputing" all of $f(X)$.*

A common goal in multi-party computation is to compute the output of some cryptographic algorithm (e.g., decryption, signing, etc.), where the secrets have been shared. Ideally, the participants to the protocol would only learn the output (e.g., the plaintext), but nothing else (including the shared secret). An important technique that applies to DL protocols is "Shamir's secret sharing in the exponents", introduced by Desmedt and Frankel [22].

For simplicity, let's start from the case of decrypting ElGamal ciphertexts with linearly shared secrets. Let $G = \langle g \rangle$ be a group of order p, let $1 \leq s \leq p$ be a secret key and let g^s be the associated public key. Decompose the secret s as

$$s = s_1 + \cdots + s_n$$

and distribute the s_i to n participants, numbered from 1 to n. Let $C = (C_1, C_2) = (g^r, g^{sr} \cdot m)$ be an ElGamal ciphertext, the goal is for the participants to compute m without anyone learning s. The protocol proceeds as follows:

1. Each participant broadcasts $C_1^{s_i} = g^{s_i r}$;
2. Each participant computes $g^{sr} = \prod g^{s_i r}$;
3. Each participant computes $m = C_2 / g^{sr}$.

Exercise 30. *Generalize the protocol above to a k-out-of-n threshold protocol where s has been shared using Shamir's secret sharing.*

Exercise 31. *Define the analogous distributed protocol for Schnorr signatures.*

It is tempting to generalize the protocol above to cryptographic group actions: decompose $s \in G$ as $s = s_1 + \cdots s_n$, then have each participant broadcast $s_i \star C_1$. However we face a problem in step 2: there is no equivalent of the product $\prod g^{s_i r}$ in the group action setting.

Unfortunately, we know no real way around this problem. What is proposed in [19] is to define a weaker form of distributed protocol where the participants act in sequence rather than in parallel. Given a ciphertext

$$C = (C_1, C_2) = (r \star x, H((sr) \star x) \oplus m),$$

let $x_0 = r \star x$. For each i from 1 to n, participant i computes $x_i = s_i \star x_{i-1}$ and passes this value to participant $i+1$. The last participant can compute $x_n = s_n \star x_{n-1} = (sr) \star x$, and thus recover the plaintext m.

It is now tempting to give as an exercise to generalize the idea above to Shamir's secret sharing, but there is one additional difficulty that calls for, at least, a hint. Being based on polynomial interpolation, Shamir's secret sharing requires the shares s_i to be elements of a ring. Here, the s_i only belong to a group. The trick is to go "in the exponents of the exponents"

Let $(G, X, \star)$ be an EGA and assume that $G = \langle g \rangle$ is cyclic of order N, then there is a (group) homomorphism

$$\begin{aligned}\mathbb{Z}/N\mathbb{Z} &\to G,\\ a &\mapsto g^a.\end{aligned}$$

Hence, we can define an (additive) group action $*$ of $\mathbb{Z}/N\mathbb{Z}$ on X by

$$a * x := g^a \star x.$$

Almost magically, all the pieces now fit in place and we can define threshold decryption using Shamir's secret sharing.

Exercise 32. *(hard?) Generalize the distributed EGA–ElGamal decryption protocol above to Shamir's secret sharing.*

Note that for this generalization to work several conditions must be met:

- $(G, X, \star)$ must absolutely be an EGA (exercise: explain why a REGA is not sufficient);
- G must be cyclic, or at least contain a large cyclic group, with a known generator g;
- The order N of g must be known, and the smallest prime divisor of N must be larger than the number n of participants.

Despite all the difficulties involved, the good news is that by solving this exercise you become a pirate: `https://www.youtube.com/watch?v=gjqbgUVZcwI`! If you're stuck, have a look at [19]. Further generalizations of multi-party protocols based on group actions appeared in [16, 4].

12 Other protocols

We're at the 19 pages mark, but we're far from having exhausted the protocols based on group actions. Explaining all of them would take us too far, so we just list some works here for completeness:

- [1] defines several protocols based on wPR-(R)EGA, including *hash proof systems* and *dual mode encryption*.
- [1] also defines a new, plausible, security assumption for (R)EGA they call *linear hidden shift*, from which they derive a *key-dependent-message-CPA-secure symmetric encryption* scheme.
- [23] instantiates *symmetric updatable encryption* from EGA (but not REGA) of known order, and discusses obstacles to instantiating *asymmetric updatable encryption*.
- [5] instantiates *ring signatures* from (R)EGA.

- **Exercise:** help me complete this list with the constructions I'm forgetting.

Despite the remarkable number of different primitives that can be instantiated from group actions, several important ones are missing, and appear to be difficult to instantiate. Let me give some examples.

Problem 33. *Instantiate* (simply) homomorphic encryption *from (R)EGA.*

Problem 34. *Instantiate* collision-resistant hash functions *from (R)EGA (i.e., a hash function where collision resistance reduces to a plausible assumption for (R)EGAs).*

Problem 35. *Instantiate* identity based *or* attribute based encryption *from (R)EGA.*

References

[1] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 411–439. Springer, Heidelberg, December 2020. `doi:10.1007/978-3-030-64834-3_14`.

[2] Paulo Barreto, Glaucio Oliveira, and Waldyr Benits. Supersingular isogeny oblivious transfer. Cryptology ePrint Archive, Report 2018/459, 2018. `https://eprint.iacr.org/2018/459`.

[3] Ward Beullens. Week 4: Signatures based on SIDH and CSIDH, 2021. URL: `https://homes.esat.kuleuven.be/~wbeullen/week4_1.pdf`.

[4] Ward Beullens, Lucas Disson, Robi Pedersen, and Frederik Vercauteren. CSI-RAShi: Distributed key generation for CSIDH. Cryptology ePrint Archive, Report 2020/1323, 2020. `https://eprint.iacr.org/2020/1323`.

[5] Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 464–492. Springer, Heidelberg, December 2020. `doi:10.1007/978-3-030-64834-3_16`.

[6] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Heidelberg, December 2019. `doi:10.1007/978-3-030-34578-5_9`.

[7] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 757–788. Springer, Heidelberg, August 2018. `doi:10.1007/978-3-319-96884-1_25`.

[8] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_13.

[9] Dan Boneh, Dmitry Kogan, and Katharine Woo. Oblivious pseudorandom functions from isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 520–550. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64834-3_18.

[10] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001. doi:10.1007/3-540-45682-1_30.

[11] Gilles Brassard and Moti Yung. One-way group actions. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 94–107. Springer, Heidelberg, August 1991. doi:10.1007/3-540-38424-3_7.

[12] Jeffrey Burdges and Luca De Feo. Delay encryption. Cryptology ePrint Archive, Report 2020/638, 2020. https://eprint.iacr.org/2020/638.

[13] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 395–427. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-03332-3_15.

[14] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 40–58. Springer, Heidelberg, August 2015. doi:10.1007/978-3-319-22174-8_3.

[15] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. https://eprint.iacr.org/2006/291.

[16] Daniele Cozzo and Nigel P. Smart. Sashimi: Cutting up CSI-FiSh secret keys to produce an actively secure distributed signing protocol. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 169–186. Springer, Heidelberg, 2020. doi:10.1007/978-3-030-44223-1_10.

[17] Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 759–789. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17659-4_26.

[18] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 248–277. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-34578-5_10.

[19] Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 187–212. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45388-6_7.

[20] Cyprien de Saint Guilhem, Emmanuela Orsini, Christophe Petit, and Nigel P. Smart. Semi-commutative masking: A framework for isogeny-based protocols, with an application to fully secure two-round isogeny-based OT. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 235–258. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-65411-5_12.

[21] Thomas Decru, Lorenz Panny, and Frederik Vercauteren. Faster SeaSign signatures through improved rejection sampling. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 271–285. Springer, Heidelberg, 2019. doi:10.1007/978-3-030-25510-7_15.

[22] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, Heidelberg, August 1990. doi:10.1007/0-387-34805-0_28.

[23] Edward Eaton, David Jao, and Chelsea Komlo. Towards post-quantum updatable public-key encryption via supersingular isogenies. Cryptology ePrint Archive, Report 2020/1593, 2020. URL: https://eprint.iacr.org/2020/1593.

[24] Edward Eaton and Douglas Stebila. The "quantum annoying" property of password-authenticated key exchange protocols. Cryptology ePrint Archive, Report 2021/696, 2021. URL: https://ia.cr/2021/696.

[25] Steven D Galbraith. *Mathematics of public key cryptography*. Cambridge University Press, 2012. URL: https://www.math.auckland.ac.nz/~sgal018/crypto-book/crypto-book.html.

[26] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113 – 3121, 2008. Applications of Algebra to Cryptography. doi:10.1016/j.dam.2007.12.010.

[27] Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online). In *2016 IEEE European Symposium*

on Security and Privacy (EuroS&P), pages 276–291. IEEE, 2016. `doi:10.1109/EuroSP.2016.30`.

[28] Takeshi Koshiba and Katsuyuki Takashima. Pairing cryptography meets isogeny: A new framework of isogenous pairing groups. Cryptology ePrint Archive, Report 2016/1138, 2016. `https://eprint.iacr.org/2016/1138`.

[29] Takeshi Koshiba and Katsuyuki Takashima. New assumptions on isogenous pairing groups with applications to attribute-based encryption. In Kwangsu Lee, editor, *ICISC 18*, volume 11396 of *LNCS*, pages 3–19. Springer, Heidelberg, November 2019. `doi:10.1007/978-3-030-12146-4_1`.

[30] Yi-Fu Lai, Steven D. Galbraith, and Cyprien de Saint Guilhem. Compact, efficient and UC-secure isogeny-based oblivious transfer. Cryptology ePrint Archive, Report 2020/1012, 2020. `https://eprint.iacr.org/2020/1012`.

[31] Ueli M. Maurer and Stefan Wolf. Lower bounds on generic algorithms in groups. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 72–84. Springer, Heidelberg, May / June 1998. `doi:10.1007/BFb0054118`.

[32] Tomoki Moriya, Hiroshi Onuki, and Tsuyoshi Takagi. SiGamal: A supersingular isogeny-based PKE and its application to a PRF. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 551–580. Springer, Heidelberg, December 2020. `doi:10.1007/978-3-030-64834-3_19`.

[33] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997. `doi:10.1109/SFCS.1997.646134`.

[34] Krzysztof Pietrzak. Simple verifiable delay functions. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 60:1–60:15. LIPIcs, January 2019. `doi:10.4230/LIPIcs.ITCS.2019.60`.

[35] Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA, 1996. URL: `https://people.csail.mit.edu/rivest/pubs/RSW96.pdf`.

[36] Alexander Rostovtsev and Anton Stolbunov. Public-Key Cryptosystem Based On Isogenies. Cryptology ePrint Archive, Report 2006/145, 2006. `https://eprint.iacr.org/2006/145`.

[37] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[38] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. `doi:10.1007/3-540-69053-0_18`.

[39] Joseph H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1992.

[40] Vanessa Vitse. Simple oblivious transfer protocols compatible with supersingular isogenies. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 56–78. Springer, Heidelberg, July 2019. `doi:10.1007/978-3-030-23696-0_4`.

[41] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407. Springer, Heidelberg, May 2019. `doi:10.1007/978-3-030-17659-4_13`.