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:

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

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 [14, § 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 $h = 0$, and*
- $e(g, h) = 1$ *for all h implies $g = 0$.*

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*

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

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., [7]). 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 [8].

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 [10, 11], 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 [6]. 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 be the restriction of ϕ to $E[N]$. Propose an algorithm to distinguish f from a random function $E[N] \to E'[N]$ using a CDH solver.*

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.*

Note that the only known way to solve CDH is to compute a discrete logarithm. When thinking about the quantum annoying property, 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 [6], 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 [3].

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

The construction above is easily extended to isogenous pairing groups. In [5] 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.*

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 [9]:

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) [12, 15], 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 let N be an integer dividing the order of E_i, 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 [5]. 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[1], 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 [5, 4] 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:

- 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) [1]. To the second belong *time-lock puzzles* (TLP) [13] and *delay encryption* (DE) [4].

[1] 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.

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 [2]. 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*.* 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 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 [4, § 3]. One notable fact about DE is that this isogeny based construction is the *only known instantiation* of DE.

Part II
Cryptographic Group Actions

coming soon...

References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.