Delay Encryption

Jeffrey Burdges[1] and Luca De Feo[2][0000−0002−9321−0773]

[1] Web 3, Switzerland
[2] IBM Research Zürich, Switzerland eurocrypt21@defeo.lu

Abstract. We introduce a new primitive named Delay Encryption, and give an efficient instantation based on isogenies of supersingular curves and pairings. Delay Encryption is related to Time-lock Puzzles and Verifiable Delay Functions, and can be roughly described as "time-lock identity based encryption". It has several applications in distributed protocols, such as sealed bid Vickrey auctions and electronic voting.
We give an instantiation of Delay Encryption by modifying Boneh and Frankiln's IBE scheme, where we replace the master secret key by a long chain of isogenies, as in the isogeny VDF of De Feo, Masson, Petit and Sanso. Similarly to the isogeny-based VDF, our Delay Encryption requires a trusted setup before parameters can be safely used; our trusted setup is identical to that of the VDF, thus the same parameters can be generated once and shared for many executions of both protocols, with possibly different delay parameters.
We also discuss several topics around delay protocols based on isogenies that were left untreated by De Feo *et al.*, namely: distributed trusted setup, watermarking, and implementation issues.

Keywords: Delay functions · Isogenies · Pairings · Supersingular elliptic curves

1 Introduction

The first appearance of *delay cryptography* was in Rivest, Shamir and Wagner's [29] *Time-lock Puzzle*, an encryption primitive where the holder of a trapdoor can encrypt (or decrypt) "fast", but where anyone not knowing the trapdoor can only decrypt (or encrypt) "slowly".

Recently, a revival of delay cryptography has been promoted by research on blockchains, in particular thanks to the introduction of *Verifiable Delay Functions (VDF)* [4]: deterministic functions f that can only be evaluated "sequentially" and "slowly", but such that verifying that $y = f(x)$ is "fast".

After their definition, VDFs quickly gained attention, prompting two independent solutions in the space of a few weeks [34,27]. Both proposals are based on repeated squaring in groups of unknown order, and are similar in spirit to Rivest *et al.*'s Time-lock Puzzle, however they use no trapdoor.

One year later, another VDF, based on a different algebraic structure, was proposed by De Feo, Masson, Petit and Sanso [17]. This VDF uses chains of supersingular isogenies as "sequential slow" functions, and pairings for efficient

verification. Interestingly, it is not known how to build a Time-lock Puzzle from isogenies; in this work we introduce a new primitive, in some respects more powerful than Time-lock Puzzles, that we are able to instantiate from isogenies.

Limitations of Time-lock Puzzles. Time-lock Puzzles allow one to "encrypt to the future", i.e., to create a puzzle π that encapsulates a message m for a set amount of time T. They have the following two properties:

- Puzzle generation is efficient: there exists an algorithm which, on input the message m and the *delay* T, generates π in time much less than T.
- Puzzle solving is predictably slow and sequential: on input π, the message m can be recovered by a circuit of depth approximately T, and no circuit of depth less than T can recover m reliably.

Time-lock Puzzles can be used to remove trusted parties from protocols, replacing them with a time-consuming puzzle solving. Prototypical applications are auctions and electronic voting, we will use auctions as a motivating example.

In a highest bidder auction, the easy solution in presence of a trusted authority is to encrypt bids to the authority, who then decrypts all the bids and selects the winner. Lacking a trusted authority, the standard solution is to divide the auction in two phases: in the *bidding phase* all bidders commit to their bids using a commitment; in the *tallying phase* bidders open their commitments, and the highest bidder wins. However, this design has one flaw in contexts where it is required that all bidders reveal their bids at the end of the auction. For example, in *Vickrey auctions*, the highest bidder wins the auction, but only pays the price of the second highest bid. If at the end of the auction some bidders refuse to open their commitment, the result of the auction may be invalid.

Time-lock Puzzles solve this problem: by having bidders encapsulate their bid in a Time-lock Puzzle, it is guaranteed that all bids can be decrypted in the tallying phase. However this solution becomes very expensive in large auctions, because one puzzle per bidder must be solved: if several thousands of bidders participate, the tallyers must strike a balance between running thousands of puzzle solving computations in parallel, and having a tallying phase that is thousands of times longer than the bidding phase. Since Time-lock Puzzles use trapdoors for puzzle generation, a potential mitigation is to have the bidders reveal their trapdoors in the tallying phase, thus speeding up decryption; however this does not help in presence of a large number of uncollaborative bidders.

An elegant solution introduced in [25] is to use *Homomorphic Time-lock Puzzles (HTLP)*, i.e., Time-lock Puzzles where the puzzles can be efficiently combined homomorphically. Using these, the tallyers can efficiently evaluate the desired tallying circuit on the unopen puzzles, and then run only a single slow puzzle-solving algorithm. Unfortunately, the only efficient HTLPs introduced in [25] are simply homomorphic (either additively or multiplicatively), and they are thus only useful for voting; fully homomorphic TLPs, which are necessary for auctions, are only known from Fully Homomorphic Encryption [9] or from Indistinguishability Obfuscation [25], and are thus unpractical.

On top of that, it can be argued that Time-lock Puzzles are not the appropriate primitive to solve the problem: why do the tallyers need to run one of two different algorithms to open the puzzles? Are trapdoors really necessary? In this work, we introduce a new primitive, *Delay Encryption*, that arguably solves the problem more straightforwardly and elegantly.

Delay Encryption. Delay Encryption is related to both Time-lock Puzzles and VDFs, however it does not seem to be subsumed by either. It can be viewed as a time-lock analogue of Identity Based Encryption, where the derivation of individual private keys is sequential and slow.

Instead of senders and receivers, Delay Encryption has a concept of *sessions*. A session is defined by a *session identifier*, which must be a hard to predict string. When a session identifier id is issued, anyone knowing id can *encrypt to the session for* id; decryption is however unfeasible without a *session key*, which is to be derived from id. The defining feature of Delay Encryption is *extraction*: the process of deriving a session key from a session identifier. Extraction must be a sequential and slow operation, designed to run in time T and no less for almost any id.

Since there are no secrets in Delay Encryption, anyone can run extraction. It is thus important that session identifiers are hard to predict, and thrown away after the first use, otherwise an attacker may precompute session keys and immediately decrypt any ciphertext to the sessions.

Delay Encryption is different from known Time-lock Puzzles in that it has no trapdoor, and from VDFs in that it provides a fast encryption, rather than just a fast verification. It has similar applications to Homomorphic Time-lock Puzzles, it is however more efficient and solves many problems more straightforwardly.

Applications of Delay Encryption. We already mentioned the two main applications of Time-lock Puzzles. We review here how Delay Encryption offers better solutions.

Vickrey auctions. Sealed bid auctions are easily implemented with standard commitments: in the bidding phase each bidder commits to its bid; later, in the tallying phase each bidder opens their commitment. However this solution is problematic when some bidders may refuse to open their commitments.

Delay Encryption provides a very natural solution: at the beginning of the auction an *auction identifier* is selected using some unpredictable and unbiased randomness, e.g., coming from a randomness beacon. After the auction identifier is published, all bidders encrypt to the auction as senders of a Delay Encryption scheme. In the meantime, anyone can start computing the auction key using the extraction functionality. When the *auction key* associated with the auction identifier is known, anyone in possession of it can decrypt all bids and determine the winner.

Electronic voting. In electronic voting it is often required that the partial tally of an election stays unknown until the end, to avoid influencing the outcome.

Delay Encryption again solves the problem elegantly: once the *election identifier* is published, all voters can cast their ballot by encrypting to it. Only after the election key is published, anyone can verify the outcome by decrypting the ballots.

Of course this idea can be combined with classical techniques for anonymity, integrity, etc.

In both applications it is evident that the session/auction/election identifier must be unpredictable and unbiased: if it is not, someone may start computing the session key before anyone else can, and thus break the delay property. Fortunately, this requirement is easily satisfied by using randomness beacons, which, conveniently, can be implemented using VDFs.

Contributions. Our main contribution is the introduction of Delay Encryption: we formally define the primitive and its security, then argue about its naturalness by relating it to other well known primitives such as IBE and VDFs.

Building on Boneh and Franklin's IBE scheme [6], and on a framework introduced in [17] for VDFs, we give an instantiation of Delay Encryption from isogeny walks in graphs of pairing friendly supersingular elliptic curves. We prove the security of our instantiation using a new assumption, related to both the Bilinear Diffie-Hellman assumption typical of pairing based protocols, and the Isogeny Shortcut assumption used for isogeny based VDFs.

Additionally, we cover some topics related to isogeny-based delay functions which apply to both our Delay Encryption and to VDFs, which were left untreated by [17]:

1. We show how to realize the trusted setup needed in all isogeny-based delay protocols in a distributed manner, and propose an efficient implementation based on a new zero-knowledge proof of isogeny knowledge —whose security we are only able to prove heuristically using a non-falsifiable assumption.
2. We show how to claim "ownership" of a delay function evaluation (aka extraction, in the Delay Encryption jargon), by attaching a "watermark" to the result of the evaluation. Watermarking can be used in distributed consensus protocols to reward the party who bears the load of evaluating the delay function.
3. We provide new elliptic curve representations and isogeny formulas optimized for the operations occurring in isogeny based delay functions. Based on these, we estimate the length of the isogeny walk needed to achieve a certain delay, and the size of the associated public parameters.

Plan. Delay Encryption is defined in Section 2, and our instantiation is given in Section 3. Each of the following sections discusses one topic related to both Delay Encryption and VDFs based on isogenies: Section 4 discusses the trusted setup, Section 5 covers watermarking, Section 6 introduces the new isogeny formulas and makes some practical considerations.

2 Definitions

Our definition of Delay Encryption uses an API similar to a Key Encapsulation Mechanism: it consists of four algorithms —Setup, $\mathsf{Extract}$, Encaps and Decaps— with the following interface.

$\mathsf{Setup}(\lambda, T) \to (\mathsf{ek}, \mathsf{pk})$**.** 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, see below.

$\mathsf{Encaps}(\mathsf{pk}, \mathsf{id}) \to (c, k)$**.** 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, which can then be used as in the applications described previously. Alternatively we could have used a PKE-like API directly, however we prefer the KEM one as it is closer to known instantiations.

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. We now give the formal definition.

The security game. It is apparent from the definitions that Delay Encryption has no secrets: after public parameters $(\mathsf{ek}, \mathsf{pk})$ are generated, anyone can run any of the algorithms. Thus, the usual notion of indistinguishability will only be defined with respect to the delay parameter T: no adversary is able to distinguish a key k from a random string in time $T - o(T)$, but anyone can in time T. Properly defining what is meant by "time" requires fixing a computation model. Here we follow the usual convention from VDFs, and assume a model of parallel computation: in this context, "time T" may mean T steps of a parallel Turing machine, or an arithmetic circuit of depth T. Crucially, we do not bound the

amount of parallelism of the Turing machine, or the breadth of the circuit, i.e., we focus on *sequential delay* functions.

We consider the following Δ-IND-CPA game. Note that the game involves no oracles, owing to the fact that the scheme has no secrets.

Precomputation. The adversary receives $(\mathsf{ek}, \mathsf{pk})$ as input, and outputs an algorithm $\mathcal{D}$.

Challenge. The challenger selects a random id and computes a key encapsulation $(c, k_0) \leftarrow \mathsf{Encaps}(\mathsf{pk}, \mathsf{id})$. It then picks a uniformly random $k_1 \in \mathcal{K}$, and a random bit $b \in \{0, 1\}$. Finally, it outputs (c, k_b, id).

Guess. Algorithm $\mathcal{D}$ is run on input (c, k_b, id). The adversary wins if $\mathcal{D}$ terminates in time less than Δ, and the output is such that $\mathcal{D}(c, k_b, \mathsf{id}) = b$.

We stress that the game is intrinsically non-adaptive, in the sense that no computation is "free" after the adversary has seen the challenge.

We say a Delay Encryption scheme is *Δ-Delay Indistinguishable under Chosen Plaintext Attacks* if any adversary running the precomputation in time $\mathrm{poly}(\lambda, T)$ has negligible advantage in winning the game. Obviously, the interesting schemes are those where $\Delta = T - o(T)$.

Remark 1. Although it would be possible to define an analogue of chosen ciphertext security for Delay Encryption, by giving algorithm $\mathcal{D}$ access to a decryption oracle for id, it is not clear what kind of real world attacks this security notion could model. Indeed, an instantaneous decryption oracle for id would go against the idea that the session key idk needed for decryption is not known to anyone before time T.

Similarly, one could imagine giving $\mathcal{D}$ access to an extraction oracle, to allow it instantaneous adaptive extraction queries after the challenge (note that in the precomputation phase the adversary is free to run polynomially many non-adaptive extractions). However it is not clear what component of a real world system could provide such instantaneous extraction in practice, since extraction is a public (and slow) operation.

2.1 Relationship with other primitives

Delay Encryption and Identity Based Encryption. Although there is no formal relationship between Identity Based Encryption (IBE) and Delay Encryption, the similarity is evident.

Recall that an IBE scheme is a public key encryption with three parties: a dealer who possesses a master private/public key pair, a receiver who has an *identity* that acts as its public key (e.g., its email address), and a sender who wants to send a message to the receiver. In IBE, the dealer runs an *extraction* on the identity to generate the receiver's secret key. The sender encrypts messages to the receiver using both the identity and the master public key. The receiver decrypts using the master public key and the private key provided by the dealer.

Delay Encryption follows the same blueprint, but has no secrets: there is no master key anymore, but only a set of public parameters $(\mathsf{ek}, \mathsf{pk})$. Receiver identities become session identifiers id: public but unpredictable. The dealer is replaced

by the public functionality $\mathsf{Extract}(\mathsf{ek}, \mathsf{id})$: sequential and slow. Senders encrypt messages to the sessions by using pk and id. After extraction has produced idk, anyone can decrypt messages "sent" to id by using idk.

The similarity with IBE is not fortuitous. Indeed, the instantiation we present next is obtained from Boneh and Franklin's IBE scheme [6], by replacing the master secret with a public, slow to evaluate, isogeny. This is analogous to the way De Feo *et al.*'s VDF [17] is obtained from the Boneh–Lynn–Shacham signature scheme [7].

The similarity with IBE will be mirrored both in the reductions we discuss next, and in the security proof of our instantiation.

Delay Encryption and Verifiable Delay Functions. Boneh and Franklin attribute to Naor the observation that IBE implies signatures. The construction is straightforward: messages are encoded to identities; to sign a message id, simply output the derived private key idk associated to it. To verify a signature $(\mathsf{id}, \mathsf{idk})$: run encapsulation to id obtaining a random (c, k), decapsulate $(\mathsf{id}, \mathsf{idk}, c)$ to obtain k', and accept the signature if $k = k'$. The signature scheme is existentially unforgeable if the IBE scheme is indistinguishable under chosen ciphertext attacks.

Precisely the same construction shows that Delay Encryption implies (sequential) Proof of Work. Furthermore, if we define *extraction soundness* as the property that adversaries have negligible chance of finding $\mathsf{idk} \neq \mathsf{idk}'$ such that

$$\mathsf{Decaps}(\mathsf{pk}, \mathsf{id}, \mathsf{idk}, c) = \mathsf{Decaps}(\mathsf{pk}, \mathsf{id}, \mathsf{idk}', c),$$

then we see that extraction sound Delay Encryption implies Verifiable Delay Functions. It is easily verified that the derived VDF is Δ-sequential if the Delay Encryption scheme is Δ-IND-CPA.

The signature scheme derived from Boneh and Franklin's IBE is equivalent to the Boneh–Lynn–Shacham scheme. Unsurprisingly, the instantiation of Delay Encryption that we give in the next section is extraction sound, and the derived VDF is equivalent to De Feo *et al.*'s VDF.

Delay Encryption and Time-lock Puzzles. Both Delay Encryption and Time-lock Puzzles permit a form of *encryption to the future*: encrypt a message now, so that it can only be decrypted at a set time in the future. There is no formal definition of Time-lock Puzzles commonly agreed upon in the literature, it is thus difficult to say what they exactly are and how they compare to Delay Encryption.

Bitansky *et al.* [3] define Time-lock Puzzles as two algorithms

- $\mathsf{Gen}(\lambda, T, s) \to Z$ that takes as input a delay parameter T and a solution $s \in \{0,1\}^\lambda$, and outputs a puzzle Z;
- $\mathsf{Solve}(Z) \to s$ that takes as input a puzzle Z and outputs the solution s;

under the constraints that Gen runs in time $\mathrm{poly}(\lambda, \log T)$ and that no algorithm computes s from Z in parallel time significantly less than T.

One might be tempted to construct a Time-lock Puzzle from Delay Encryption by defining Gen as follows:

1. Compute $(\mathsf{ek}, \mathsf{pk}) \leftarrow \mathsf{Setup}(\lambda, T)$;
2. Sample a random $\mathsf{id} \in \{0,1\}^\lambda$;
3. Compute $(c, k) \leftarrow \mathsf{Encaps}(\mathsf{pk}, \mathsf{id})$;
4. Compute $m = E_k(s)$;
5. Return $(\mathsf{ek}, \mathsf{pk}, \mathsf{id}, c, m)$;

where E_k is a symmetric encryption scheme. Then Solve is naturally defined as

1. Compute $\mathsf{idk} \leftarrow \mathsf{Extract}(\mathsf{ek}, \mathsf{id})$;
2. Compute $k \leftarrow \mathsf{Decaps}(\mathsf{pk}, \mathsf{id}, \mathsf{idk}, c)$;
3. Return $s = D_k(m)$;

where D_k is the decryption routine associated to E_k.

However this fails to define a Time-lock Puzzle in the sense above, because Setup can take time $\mathrm{poly}(\lambda, T)$ instead of $\mathrm{poly}(\lambda, \log T)$. If we take Setup out of Gen, though, we obtain something very similar to what Bitansky *et al.* call Time-lock Puzzles *with pre-processing*, albeit slightly weaker.[3]

We see no technical obstruction to having Setup run in time $\mathrm{poly}(\lambda, \log T)$, and thus being a strictly stronger primitive than Time-lock Puzzles. However our instantiation does not satisfy this stronger notion of Delay Encryption, and, lacking any other candidate, we prefer to keep our definitions steeping in reality.

To summarize, Delay Encryption is a natural analogue of Identity Based Encryption in the world of time delay cryptography. It requires Proofs of Work to exist, and a mild strengthening of it (which we are able to instantiate) implies Verifiable Delay Functions. It also implies a weak form of Time-lock Puzzles, and a strengthening of it (of which we know no instantiation) implies standard Time-lock Puzzles. At the same time, no dependency is known between Time-lock Puzzles and Verifiable Delay Functions, indicating that Delay Encryption is possibly a stronger primitive than both.

3 Delay Encryption from isogenies and pairings

We instantiate Delay Encryption from the same framework De Feo, Masson, Petit and Sanso used to instantiate Verifiable Delay Functions [17]. We briefly recall it here for completeness.

An elliptic curve E over a finite field $\mathbb{F}_{p^n}$ is said to be supersingular if the trace of its Frobenius endomorphism is divisible by p, i.e., if $\#E(\mathbb{F}_{p^n}) = 1 \mod p$. Over the algebraic closure of $\mathbb{F}_p$, there is only a finite number of isomorphism classes of supersingular curves, and every class contains a curve defined over $\mathbb{F}_{p^2}$.

We will only work with supersingular curves $E/\mathbb{F}_{p^2}$ whose group of $\mathbb{F}_{p^2}$-rational points is isomorphic to $(\mathbb{Z}/(p+1)\mathbb{Z})^2$. For these curves, if N is a divisor of $p+1$, we will denote by $E[N]$ the subgroup of $\mathbb{F}_{p^2}$-rational points of N-torsion, which is isomorphic to $(\mathbb{Z}/N\mathbb{Z})^2$. We will write $E[N]^\circ$ for the subset of points

[3] Bitansky *et al.* require pre-processing to run in sequential time $T \cdot \mathrm{poly}(\lambda)$, but parallel time only $\mathrm{poly}(\lambda, \log T)$.

in $E[N]$ of order exactly N; when N is prime, this is simply a shorthand for $E[N] \setminus \{\mathcal{O}\}$.

However, among these curves some will be curves $E/\mathbb{F}_p$ defined over $\mathbb{F}_p$, seen as curves over $\mathbb{F}_{p^2}$ (in algebraic jargon, with scalars extended from $\mathbb{F}_p$ to $\mathbb{F}_{p^2}$). For this special case, if N is an odd divisor of $p+1$, the $\mathbb{F}_{p^2}$-rational torsion subgroup $E[N]$ contains two distinguished subgroups: the subgroup $E[N] \cap E(\mathbb{F}_p)$ of points of order N defined over $\mathbb{F}_p$, which we also denote by $E[(N, \pi - 1)]$; and the subgroup of points of order N not in $E(\mathbb{F}_p)$, but with x-coordinate in $\mathbb{F}_p$, which we denote by $E[(N, \pi + 1)]$. Again, we write $E[(N, \pi - 1)]^\circ$ and $E[(N, \pi + 1)]^\circ$ for the subsets of points of order exactly N.

An isogeny is a group morphism of elliptic curves with finite kernel. In particular, isogenies preserve supersingularity. Isogenies can be represented by ratios of polynomials, and, like polynomials, have a *degree*. Isogenies of degree ℓ are also called ℓ-isogenies; the degree is multiplicative with respect to composition, thus $\deg \phi \circ \psi = \deg \phi \cdot \deg \psi$. The degree is an important invariant of isogenies, roughly measuring the amount of information needed to represent them.

An isogeny graph is a graph whose vertices are isomorphism classes of elliptic curves, and whose edges are isogenies, under some restrictions. Isogeny-based cryptography mainly uses two types of isogeny graphs:

- The *full supersingular graph* of (the algebraic closure of) $\mathbb{F}_p$, whose vertices are all isomorphism classes of supersingular curves over $\mathbb{F}_{p^2}$, and whose edges are all isogenies of a prime degree ℓ; typically $\ell = 2, 3$.
- The $\mathbb{F}_p$*-restricted supersingular graph*, or *supersingular CM graph* of $\mathbb{F}_p$, whose vertices are all $\mathbb{F}_p$-isomorphism classes of supersingular curves over $\mathbb{F}_p$, and whose edges are ℓ-isogenies for all primes ℓ up to some bound; typically $\ell \lessapprox \lambda \log \lambda$, where λ is the security parameter.

Any ℓ-isogeny $\phi : E \to E'$ has a unique *dual* ℓ-isogeny $\hat{\phi} : E' \to E$ such that

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

for any integer N and any points $P \in E[N]$, $Q \in E'[N]$, where e_N is the Weil pairing on E, and e'_N the one on E'. The same equation, with the same $\hat{\phi}$, also holds for any other known pairing, such as the Tate and Ate pairings.

The framework of De Feo *et al.* uses chains of small degree isogenies as delay functions, and the pairing equation (1) as an efficient means to verify the computation. Formally, they propose two related instantiations of VDF, following the same pattern: they both use the same base field $\mathbb{F}_p$, where p is a prime of the form $p+1 = N \cdot f$ with N prime, chosen so that discrete logarithms in the group of N-th roots of unity in $\mathbb{F}_{p^2}$ (the target group G_T of the pairing) are hard (i.e., $N \approx 2^{2\lambda}$ and $p \sim 2^{\lambda^3}$). They have a common trusted setup, independent of the delay parameter, and the usual functionalities of a VDF:

Trusted setup selects a random supersingular elliptic curve E over $\mathbb{F}_p$.

Setup takes as input p, N, E, a delay parameter T, and performs a walk in an ℓ-isogeny graph to produce a degree ℓ^T isogeny $\phi : E \to E'$.
It also computes a point $P \in E(\mathbb{F}_p)$ of order N. It outputs $\phi, E', P, \phi(P)$.

Evaluation takes as input a random point $Q \in E'[N]$ and outputs $\hat{\phi}(Q)$.

Verification uses Eq. (1) to check that the value output by evaluation is $\hat{\phi}(Q)$ as claimed.

The two variants only differ in the way the isogeny walk is set up, and in minor details of the verification; these differences will be irrelevant to us.

The delay property of this VDF rests, roughly speaking, on the assumption that a chain of T isogenies of small prime degree ℓ cannot be computed more efficiently than by going through each of the isogenies one at a time, sequentially. The case $\ell = 2$ is very similar to repeated squaring in groups of unknown order as used by other VDFs [34,27] and Time-lock Puzzles [29]: in groups, one iterates T times the function $x \mapsto x^2$, a polynomial of degree 2; in isogeny graphs, one iterates rational fractions of degree 2. See Section 6 for more details.

It is important to remark that both setup and evaluation in these VDFs are "slow" algorithms, indeed both need to evaluate an isogeny chain (either ϕ, or $\hat{\phi}$) at one input point of order N; this is in stark contrast with VDFs based on groups of unknown order, where the setup, and thus its complexity, is independent of the delay parameter T.

3.1 Instantiation

The isogeny-based VDF of De Feo *et al.* can be understood as a modification on the Boneh–Lynn–Shacham [7] signature scheme, where the secret key is replaced by a long chain of isogenies: signing becomes a "slow" operation and thus realizes the evaluation function, whereas verification stays efficient.

Similarly, we obtain a Delay Encryption scheme by modifying the IBE scheme of Boneh and Franklin [6]: the master secret is replaced by a long chain of isogenies, while session identifiers play the role of identities, so that producing the decryption key for a given identity becomes a slow operation.

Concretely, setup is identical to that of the VDF. A prime of the form $p = 4 \cdot N \cdot f - 1$ is fixed according to the security parameter, then setup is actually split into two algorithms: a $\mathsf{TrustedSetup}$ independent of the delay parameter T and reusable for arbitrarily many untrusted setups, and a Setup which depends on T.

$\mathsf{TrustedSetup}(\lambda)$**.** Generate a nearly uniformly random supersingular curve $E/\mathbb{F}_p$ by starting from the curve $y^2 = x^3 + x$ and performing a random walk in the $\mathbb{F}_p$-restricted supersingular graph. Output E.

$\mathsf{Setup}(E, T)$**.**

1. Perform an ℓ-isogeny walk $\phi : E \to E'$ of length T;
2. Select a random point $P \in E(\mathbb{F}_p)$ of order N, and compute $\phi(P)$;
3. Output $\mathsf{ek} := (E', \phi)$ and $\mathsf{pk} := (E', P, \phi(P))$.

We stress that known homomorphic Time-lock Puzzles [25] also require a one-shot trusted setup. Furthermore, unlike constructions based on RSA groups,

there is no evidence that trusted setup is unavoidable for isogeny-based delay functions, and indeed removing this trusted setup is an active area of research [12,24].

The isogeny chain ϕ in Setup can be generated by any of the two methods proposed by De Feo *et al.*, the difference will be immaterial for Delay Encryption; as discussed in [17], a (deterministic) walk limited to curves and isogenies defined over $\mathbb{F}_p$ will be more efficient, however a generic (pseudorandom) walk over $\mathbb{F}_{p^2}$ will offer some partial protection against quantum attacks.

Before defining the other routines, we need two hash functions. The first, $H_1 : \{0,1\}^\lambda \to E'[N]^\circ$, will be used to hash session identifiers to points of order N in $E'/\mathbb{F}_{p^2}$ (although the curve E' may be defined over $\mathbb{F}_p$). The second, $H_2 : \mathbb{F}_{p^2} \to \{0,1\}^\lambda$, will be a key derivation function.

$\mathsf{Extract}(E, E', \phi, \mathsf{id})$.
1. Let $Q = H_1(\mathsf{id})$;
2. Output $\hat{\phi}(Q)$.

$\mathsf{Encaps}(E, E', P, \phi(P), \mathsf{id})$.
1. Select a uniformly random $r \in \mathbb{Z}/N\mathbb{Z}$;
2. Let $Q = H_1(\mathsf{id})$;
3. Let $k = e'_N(\phi(P), Q)^r$;
4. Output $(rP, H_2(k))$.

$\mathsf{Decaps}(E, E', \hat{\phi}(Q), rP)$.
1. Let $k = e_N(rP, \hat{\phi}(Q))$.
2. Output $H_2(k)$.

Correctness of the scheme follows immediately from Eq. (1) and the bilinearity of the pairing.

Remark 2. Notice that two hashed identities Q, Q' such that $Q - Q' \in \langle P \rangle$ are equivalent for encapsulation and decapsulation purposes, and thus an adversary only needs to compute the image of one of them under $\hat{\phi}$. However, if we model H_1 as a random oracle, the probability of two identities colliding remains negligible (about $1/N$).

Alternatively, if E' is defined over $\mathbb{F}_p$, one can restrict the image of H_1 to $E'[(N, \pi + 1)]$, like in [17].

3.2 Security

To prove security of their VDF schemes, De Feo *et al.* defined the following *isogeny shortcut game*:

Precomputation. The adversary receives N, p, E, E', ϕ, and outputs an algorithm $\mathcal{S}$ (in time $\mathrm{poly}(\lambda, T)$).
Challenge. The challenger outputs a uniformly random $Q \in E'[N]$.
Guess. The algorithm $\mathcal{S}$ is run on input Q. The adversary wins if $\mathcal{S}$ terminates in time less than Δ, and $\mathcal{S}(Q) = \hat{\phi}(Q)$.

However, it is clear that the Δ-hardness of this game is insufficient to prove Δ-IND-CPA security of our Delay Encryption scheme. Indeed, while the hardness of the isogeny shortcut obviously guarantees that the output of $\mathsf{Extract}$ cannot be computed in time less than Δ, there is at least one other way to decapsulate a ciphertext rP, which consists in evaluating $\phi(rP)$ and computing $k = e'_N(\phi(rP), Q)$. Computing $\phi(rP)$ is expected to be at least as "slow" as computing $\hat{\phi}(Q)$, however this fact is not captured by the isogeny shortcut game.

Instead, we define a new security assumption, analogous to the Bilinear Diffie-Hellman assumption used in standard pairing-based protocols. The *bilinear isogeny shortcut game* is defined as follows:

Precomputation. The adversary receives p, N, E, E', ϕ, and outputs an algorithm $\mathcal{S}$.

Challenge. The challenger outputs uniformly random $R \in E[(N, \pi - 1)]$ and $Q \in E'[N]$.

Guess. Algorithm $\mathcal{S}$ is run on (R, Q). The adversary wins if $\mathcal{S}$ outputs $\mathcal{S}(R, Q) = e'_N(\phi(R), Q) = e_N(R, \hat{\phi}(Q))$.

We say that the bilinear isogeny shortcut game is Δ-hard if no adversary running the precomputation in time $\mathrm{poly}(\lambda, T)$ produces an algorithm $\mathcal{S}$ that wins in time less than Δ with non-negligible probability. The reduction to Δ-IND-CPA of our Delay Encryption scheme closely follows the proof of security of Boneh and Franklin's IBE scheme.

Theorem 1. *The Delay Encryption scheme presented above is Δ-IND-CPA secure, assuming the Δ'-hardness of the bilinear isogeny shortcut game, with $\Delta \in \Delta' - o(\Delta')$, when H_1 and H_2 are modeled as random oracles.*

Concretely, suppose there is a Δ-IND-CPA adversary $\mathcal{A}$ with advantage ϵ and complexity $\mathrm{poly}(\lambda, T)$, making q queries to H_2 (including the queries made by the sub-algorithm $\mathcal{D}$). Then there is a $\mathrm{poly}(\lambda, T)$ algorithm $\mathcal{B}$ that wins the bilinear isogeny shortcut game with probability at least $2\epsilon/q$ and delay $\Delta' = \Delta + q \cdot \mathrm{poly}(\lambda)$.

Proof. In the precomputation phase, when $\mathcal{B}$ receives the parameters p, N, E, E', ϕ, it draws a random $P \in E(\mathbb{F}_p)$ of order N, and evaluates $\phi(P)$. It then passes $p, N, E, E', \phi, P, \phi(P)$ to $\mathcal{A}$ for its own precomputation phase. Whenever $\mathcal{A}$ makes calls to H_1 or H_2, algorithm $\mathcal{B}$ checks whether the input has already been requested, in which case it responds with the same answer previously given, otherwise it responds with a uniformly sampled output and records the query.

When $\mathcal{A}$ requests its challenge, $\mathcal{B}$ does the same, receiving $R \in E[(N, \pi - 1)]$ and $Q \in E'[N]$. If R or Q is the point at infinity, it outputs 1 and terminates. Otherwise it draws a random string $s \in \{0,1\}^\lambda$, a random $\mathsf{id} \in \{0,1\}^\lambda$ that was not already queried to H_1, it programs the random oracle so that $H_1(\mathsf{id}) = Q$, and challenges $\mathcal{A}$ with the tuple (R, s, id).

During the guessing phase, whenever $\mathcal{A}$ (actually, $\mathcal{D}$) makes a call to H_1 or H_2, algorithm $\mathcal{B}$ (actually, $\mathcal{S}$) responds as before. Finally, when $\mathcal{D}$ outputs its guess, $\mathcal{S}$ simply returns a random entry among those that were queried to H_2.

Let $\mathcal{H}$ be the event that $\mathcal{A}$ (or $\mathcal{D}$) queries H_2 on input $e_N(R, \hat{\phi}(Q))$. We prove that $\Pr(\mathcal{H}) \geq 2\epsilon$, which immediately gives the claim of the theorem. To this end, we first prove that $\Pr(\mathcal{H})$ in the simulation is equal to $\Pr(\mathcal{H})$ in the real attack; then we prove that $\Pr(\mathcal{H}) \geq 2\epsilon$ in the real attack.

To prove the first claim, it suffices to show that the simulation is indistinguishable from the real world for $\mathcal{A}$. Indeed, public parameters are distributed identically to a Delay Encryption scheme, and the point R that is part of the challenge is necessarily a multiple of P, since $E[(N, \pi - 1)]$ is cyclic. The proof that the two probabilities are equal, then proceeds as in [6, Lemma 4.3, Claim 1].

The proof that $\Pr(\mathcal{H}) \geq 2\epsilon$ is identical to [6, Lemma 4.3, Claim 2]. This proves the part of the statement on the winning probability of $\mathcal{B}$.

If Algorithm $\mathcal{D}$ runs in time less than Δ, algorithm $\mathcal{S}$ runs in the same time, plus the time necessary for drawing the random string s and for answering queries to H_2. Depending on the computational model, a lookup in the table for H_2 can take anywhere from $O(1)$ (e.g., RAM model) to $O(q)$ (e.g., Turing machine). We err on the safe side, and estimate that $\mathcal{S}$ runs in time less than $\Delta + q \cdot \mathrm{poly}(\lambda)$.

3.3 Known Attacks

We now shift our attention to attacks. As discussed in [17], there are three types of known attacks: *shortcut* attacks, discrete logarithm attacks, and attacks on the computation.

Parameters for a Delay Encryption scheme must be chosen so that all known attacks have exponential difficulty in the security parameter λ. Given that (total) attacks successfully compute decapsulation in exponential time in λ, it is evident that the delay parameter T must grow at most subexponentially in λ.

Shortcut attacks aim at computing a shorter path $\psi : E \to E'$ in the isogeny graph from the knowledge of $\phi : E \to E'$. The name should not be confused with the isogeny shortcut game described above, as shortcut attacks are only one of the possible ways to beat the game.

De Feo *et al.* show that shortcut attacks are possible when the endomorphism ring of at least one of E or E' is known. Indeed, in this case, the isogeny ϕ can be translated to an ideal class in the endomorphism ring, then smoothing techniques similar to [22] let us convert the ideal to one of smaller norm, and finally to an isogeny $\psi : E \to E'$ of smaller degree.

The only way out of these attacks is to select the starting curve E as a uniformly random supersingular curve over $\mathbb{F}_p$, then no efficient algorithm is known to compute $\mathrm{End}(E)$, nor $\mathrm{End}(E')$. Unfortunately, the only way we currently know to sample nearly uniformly in the supersingular class over $\mathbb{F}_p$, is, paraphrasing [20], to choose the endomorphism ring first and then compute E given $\mathrm{End}(E)$.

Thus, the solution put forth in [17] is to generate the starting curve E via a trusted setup that first selects $\mathrm{End}(E)$, and then outputs E and throws away the information about its endomorphism ring. We stress that, given a random

supersingular curve E, computing $\mathrm{End}(E)$ is a well known hard problem, upon which almost most of isogeny-based cryptography is founded. We explain in the next section how to mitigate the inconvenience of having a trusted setup, using a distributed protocol.

As stressed in [17], there is no evidence that "hashing" in the supersingular class, i.e., sampling nearly uniformly without gaining knowledge of the endomorphism ring, should be a hard problem. But there is no evidence it should be easy either, and several attempts have failed already [12,24].

Another possibility hinted at in [17] would be to generate ordinary pairing friendly curves with large isogeny class, as the shortcut attack is then thwarted by the difficulty of computing the order of the class group of the endomorphism ring. However finding such curves possibly seems an even harder problem than hashing to the supersingular class.

Discrete logarithm attacks compute $\hat{\phi}(Q)$ by directly solving the pairing equation (1). In our case, we can even directly attack the key encapsulation. Indeed, knowing rP, we obtain r through a discrete logarithm, and then compute $k = e'_N(\phi(P), Q)^r$.

Thanks to the efficiently computable pairing, the discrete logarithm can actually be solved in $\mathbb{F}_{p^2}$, which justifies taking p, N large enough to resist finite field discrete logarithm computations. Obviously, this also shows that our scheme is easily broken by quantum computers. See [17], however, for a discussion of how a setup with pseudo-random walks over $\mathbb{F}_{p^2}$ resists quantum attacks in a world where quantum computers are available, but much slower than classical ones.

Attacks on the computation do not seek to deviate from the description of the protocol, but simply try to speed up Extract beyond the way officially prescribed by the scheme. In this sort of attacks, the adversary may be given more resources than the legitimate user: for example, it may be allowed a very large precomputation, or it may dispose of an unbounded amount of parallelism, or it may have access to an architecture not available to the user (e.g., a quantum computer).

These attacks are the most challenging to analyze, because standard complexity-theoretical techniques are of little help here. On some level, this goal is unachievable: given a sufficiently abstract computational model, and a sufficiently powerful adversary, any scheme is broken. For example, an adversary may precompute all possible pairs $(Q, \hat{\phi}(Q))$ and store them in a $O(1)$-accessible RAM, then extraction amounts to a table lookup. However, such an adversary with exponential precomputation, exponential storage, and constant time RAM is easily dismissed as unreasonable. More subtle trade-offs between precomputation, storage and efficiency can be obtained, like for example RNS-based techniques to attack group-based VDFs [1]. However the real impact of these theoretical algorithms has yet to be determined.

In practice, a pragmatic approach to address attacks on the computation is to massively invest in highly specialized hardware development to evaluate the "sequential and slow" function quickly, and then produce the best designs at scale, so that they are available to anyone who wants to run the extraction. This

is the philosophy of the competitions organized by Ethereum [33] and Chia [21], targeting, respectively, the RSA based VDF and the class group based VDF.

We explore this topic more in detail in Section 6.

4 Distributed trusted setup

Trusted setup is an obvious annoyance to distributed protocols. A way to mitigate this negative impact is to distribute trust over several participants, ensuring through a multi-party computation that, if at least one participant is honest, then the setup can be trusted.

Ethereum is notoriously investing in the RSA-based VDF with Wesolowski's proof [33,34], which is known to require a trusted setup. To generate parameters, the Ethereum network will need to run a distributed RSA modulus generation, for which all available techniques essentially trace back to the work of Boneh and Franklin [5].

Distributed RSA modulus generation is notoriously a difficult task: the cost is relatively high, scales badly with the number of participants, and the attempts at optimizing it have repeatedly led to subtle and powerful attacks [31,32]. Worse still, specialized hardware for the delay function must be designed specifically for the generated modulus, which means that little design can be done prior to the distributed generation, and that if the distributed generation is then found to be rigged, a new round of distributed-generation-then-design is needed.

On the contrary, distributed parameter generation for our Delay Encryption candidate, or for the isogeny based VDF, is extremely easy. The participants start from a well known supersingular curve with known endomorphism ring, e.g., $E_0 : y^2 = x^3 - x$, and repeat, each at its own turn, the following steps:

1. Participant i checks all zero-knowledge proofs published by participants that preceded them;
2. They perform a pseudorandom isogeny walk $\psi_i : E_{i-1} \to E_i$ of length $c \log(p)$ in the $\mathbb{F}_p$-restricted supersingular graph;
3. They publish E_i, and a zero-knowledge proof that they know an isogeny $\psi : E_{i-1} \to E_i$.

The constant c is to be determined as a function of the expansion properties of the isogeny graph, and is meant to be large enough to ensure nearly uniform mixing of the walk. In practice, this constant is usually small, say $c < 10$, implying that each participant needs to evaluate a few thousands isogenies, a computation that is expected to take in the order of seconds [11].

The setup is clearly secure as long as at least one participant is honest. Indeed it is well known that computing a path from E_i to E_0 is equivalent to computing the endomorphism ring of E_i [22,18], and, since E_i is nearly uniformly distributed in the supersingular graph, the dishonest participants have no advantage in solving this problem compared to a generic attacker.

This distributed computation scales linearly with the number of participants, each participant needing to check the proofs of the previous ones. It can be left

running for a long period of time, allowing many participants to contribute trust without any need for prior registration. More importantly, it is *updatable*, meaning that after the distributed generation is complete, the final curve E can be used as the starting point for a new distributed trusted setup. This way the trusted setup can be updated regularly, building upon the trust accumulated in previous distributed generations.

Compared with the trusted setup for RSA, the outcome of the setup is much less critical for the design of hardware. Indeed, the primes p, N can be publicly chosen in advance, and hardware can be designed for them before the trusted setup is performed. The trusted curve E only impacts the first few steps of the "slow" isogeny walk $\phi : E \to E'$ generated by the untrusted setup, and can easily be integrated in the hardware design at a later stage.

4.1 Proofs of isogeny knowledge

We take a closer look at the last step each participant takes in the trusted setup: the proof of isogeny knowledge. Ignoring zero-knowledge temporarily, Eq. (1) already provides a proof of knowledge of a non-trivial relation between E_{i-1} and E_i. Let F be a deterministic function which takes as input a pair of curves E_i, E_j and outputs a pair of points in $E_i[(N, \pi - 1)]^\circ \times E_j[(N, \pi + 1)]^\circ$. Also let e_N^i denote the Weil pairing on E_i. The proof proceeds as follows

1. Map (E_{i-1}, E_i) to a pair of points $(P, Q) \leftarrow F(E_{i-1}, E_i)$;
2. Choose a random $r \in (\mathbb{Z}/N\mathbb{Z})^\times$,
3. Publish $(R, S) \leftarrow (r\psi_i(P), r\hat{\psi}_i(Q))$.

Then verification simply consists of:

1. Compute $(P, Q) \leftarrow F(E_{i-1}, E_i)$,
2. Check that $R \in E_i[(N, \pi - 1)]^\circ$ and $S \in E_{i-1}[(N, \pi + 1)]^\circ$;
3. Check that $e_N^i(R, Q) = e_N^{i-1}(P, S)$.

This proof is compact, requiring only four elements of $\mathbb{F}_p$, and efficient because computing $\psi_i(P), \hat{\psi}_i(Q)$ only adds a small overhead to the computation of ψ_i, and verification takes essentially two pairing computations. Note that the restriction in step 2 of the verification implies that for any R there exists a unique S satisfying the equation in step 3, and *vice versa*.

Remark 3. While we believe that an adversary not knowing an isogeny from E_{i-1} to E_i has a negligible probability of convincing a verifier in the protocol above, it is not clear what kind of knowledge is *exactly* proved by it. Ideally, we would like to prove that, given an algorithm that passes verification with non negligible probability, one can extract a description of some isogeny $\psi' : E_{i-1} \to E_i$.

However, no such algorithm is currently known. Related problems have been studied in the context of cryptanalyses of SIDH, under the name of "torsion point attacks" [26,30,23], however these algorithms crucially rely on the knowledge of the endomorphism ring of E_{i-1}, something we cannot exploit here.

The only way out is apparently to define a non-falsifiable "knowledge of isogeny" assumption, which would tautologically state that the protocol above is indeed a proof of knowledge of an isogeny. We defer investigation of this type of assumptions to future work.

As stated, the proof above is clearly not zero-knowledge, because the values $r\psi_i(P)$ and $r\hat{\psi}_i(Q)$ reveal a considerable amount of additional information on ψ_i. To turn the proof into a zero-knowledge one, we use Pedersen commitments to mask $r\psi_i(P)$ and $r\hat{\psi}_i(Q)$, then prove their knowledge using standard Schnorr-like proofs of knowledge.

Let F' be a function with same domain and range as F (possibly $F' = F$), and let H be a cryptographic hash function with values in $\mathbb{Z}/N\mathbb{Z}$.

We compute $P', Q' \leftarrow F'(E_i, E_{i-1})$ and choose $x, y \in (\mathbb{Z}/N\mathbb{Z})^\times$ secret, then we publish a NIZK proof for (X, Y) satisfying

$$e_N^i(X, Q)e_N^{i-1}(P, Q')^y = e_N^{i-1}(P, Y)e_N^i(P', Q)^x.$$

More precisely, we publish:

- two Pedersen commitments $X = xP' + r\psi_i(P) \in E_i[(N, \pi - 1)]$ and $Y = yQ' + r\hat{\psi}_i(Q) \in E_{i-1}[(N, \pi + 1)]$,
- two "public keys" $Y' = e_N^{i-1}(P, Q')^y$ and $X' = e_N^i(P', Q)^x$, and
- a Schnorr-like proof of knowledge (c, s_x, s_y) for x of X' and y of Y', where:
  - $s_x = k - xc$ and $s_y = k - yc$ for a randomly chosen $k \in (\mathbb{Z}/N\mathbb{Z})^\times$, and
  - $c = H(e_N^{i-1}(P, Q')\|e_N^i(P', Q)\|X'\|Y'\|e_N^i(P', Q)^k\|e_N^{i-1}(P, Q')^k)$.

At this point, our verifier now checks

- that $X \in E_i[(N, \pi - 1)]$ and $Y \in E_{i-1}[(N, \pi + 1)]$,
- that $e_N^i(X, Q)Y' = e_N^{i-1}(P, Y)X'$,
- non-triviality $X' \neq e_N^i(X, Q)$ and $Y' \neq e_N^{i-1}(P, Y)$ of the commitments, and
- the proofs of knowledge s_x, s_y using

$$c = H(e_N^{i-1}(P, Q')\|e_N^i(P', Q)\|X'\|Y'\|(X')^c e_N^i(P', Q)^{s_x}\|(Y')^c e_N^{i-1}(P, Q')^{s_y}).$$

In this, we ask verifiers to compute four pairings, which only doubles the verifier time.

The following lemma shows that this is a NIZK proof for the same statement that was proven in the simple protocol revealing $r\psi_i(P)$ and $r\hat{\psi}_i(Q)$, and thus it is a NIZK proof of isogeny knowledge, if we accept the non-falsifiable assumption mentioned in Remark 3.

Lemma 1. *Let E_{i-1}, E_i be a pair of isogenous elliptic curves, let $(P, Q) = F(E_{i-1}, E_i)$. The protocol above is a NIZK proof of knowledge of a pair (R, S) of points such that $e_N^i(R, Q) = e_N^{i-1}(P, S) \neq 1$, assuming CDH in the target group $G_T \subset \mathbb{F}_{p^2}$, and modeling H as a random oracle.*

Proof. To simulate the proof, it is enough to choose X, Y, and X' at random, set $Y' = X' e_N^{i-1}(P,Y)/e_N^i(X,Q)$, then use the simulator of the Schnorr proof to simulate knowledge of the discrete logarithms of X' and Y'.

To extract (R,S) from a prover, we start by using the Schnorr extractor to get x and y. Then, by hypothesis

$$e_N^i(X - xP', Q) = \frac{e_N^i(X,Q)}{X'} = \frac{e_N^{i-1}(P,Y)}{Y'} = e_N^{i-1}(P, Y - yQ') \neq 1,$$

thus $R = X - xP'$ and $S = Y - yQ$ is the solution we were looking for.

Since the Schnorr proof is proven secure under CDH in the ROM, the same hypotheses carry over.

For completeness, we also mention some other tools with which one might prove knowledge of this isogeny in zero knowledge, although none seem to be competitive with the technique above.

First, there exists a rapidly expanding SNARK toolbox from which one could perform $\mathbb{F}_p$ arithmetic inside the SNARK to check the verification of the second and third conditions directly. As instantiating the delay function imposes restrictions on p, one cannot necessarily select p using the Cocks-Pinch method to provide a pairing friendly elliptic curve with group order p, like in [8]. There are optimisations for arithmetic in arbitrary $\mathbb{F}_p$ however, especially using polynomial commitments, like in [19].

Second, there are well known post-quantum isogeny-based proofs:

SIDH-style proofs [15] are very inconvenient, because they require primes of a specific form, and severely limit the length of pseudo-random walks. On top of that, they are very inefficient, and do not have perfect zero-knowledge.

SeaSign-style proofs [14] have sizes in the hundreds of kilobytes, and their generation and verification are extremely slow (dozens of hours). Note that several of the optimizations used for signatures, including the class group order precomputation of CSI-FiSh [2], are not available in this context. More research on the optimization of SeaSign-style proofs for this specific context would be welcome.

SQISign-style proofs [16] are not compatible with our setting, because they require knowledge of the endomorphism rings.

5 Watermarking

When delay functions are used in distributed consensus protocols, it is common to want to reward participants who spend resources to evaluate the function. For example, in the auction application the participants who compute the session key may receive a percentage on the sale to compensate for the cost of running the extraction routine.

This raises the question of how to prove that some participant did run the computation, rather than simply steal the public output from someone else.

In the context of VDFs based on groups of unknown order, Wesolowski [34] introduced the concept of *proof watermarking*. The output of these VDFs consists of two parts: a delay function output and a proof. Wesolowski's idea is to attach to the proof a watermark based on the identity of the evaluator, so that anyone verifying the output immediately associates it to the legitimate participant. Since producing the proof costs at least as much as evaluating the delay function, a usurper who would like to claim an output for themself would need to do an amount of work comparable to legitimately evaluating the delay function, which strongly reduces the incentive for usurpation.

In the context of isogeny based VDFs, or of extraction in Delay Encryption, proof watermarking is a meaningless concept, because there is simply no proof to watermark. Nevertheless, it is possible to produce a watermark next to the output of the delay function, giving evidence that the owner of the watermark spent an amount of effort comparable to legitimately computing the output. The idea is to publish a *mid-point* update on the progress of the evaluation, and attach this mid-point to the identity of the evaluator.

Concretely, given parameters $\phi : E \to E'$ and $(P, \phi(P))$, the isogeny walk is split into two halves of equal size $\phi_1 : E \to E_{\text{mid}}$ and $\phi_2 : E_{\text{mid}} \to E'$ so that $\phi = \phi_2 \circ \phi_1$, and $\phi_1(P)$ is added to the public parameters. Each evaluator then generates a secret key $s \in \mathbb{Z}/N\mathbb{Z}$ and a public key $S = s\phi(P)$. When evaluating $\hat{\phi} = \hat{\phi}_1 \circ \hat{\phi}_2$ at a point $Q \in E'[N]$, the evaluator:

1. Computes $Q_{\text{mid}} = \hat{\phi}_2(Q)$,
2. Computes and publishes sQ_{mid},
3. Finishes off the computation by computing $\hat{\phi}(Q) = \hat{\phi}_1(Q_{\text{mid}})$.

A watermark can then be verified by checking that

$$e_N^{\text{mid}}(\phi_1(P), sQ_{\text{mid}}) = e'_N(S, Q).$$

Interestingly, this proof is *blind*, meaning that it can be verified even before the work is finished.

Given $\hat{\phi}(Q)$, a usurper wanting to claim the computation for themselves would need to either start from Q and compute $\hat{\phi}_2(Q)$, or start from $\hat{\phi}(Q)$ and compute $\frac{\phi_1(\hat{\phi}(Q))}{\deg \phi_1}$. Either way, they would perform at least half as much work as if they had legitimately evaluated the function.

This scheme is easily generalized to several equally spaced mid-points along the isogeny evaluation chain: if the isogeny is split into n pieces of equal size, a usurper would need to do at least $(n-1)/n$ times as much work as a legitimate evaluator, thus linearly decreasing the incentive for usurpation.

It is possible, nevertheless, for a usurper to target a specific evaluator, by generating a random $u \in \mathbb{Z}/N\mathbb{Z}$, and choosing $us\phi_1(P)$ as public key. Then, any proof sQ_{mid} for the legitimate evaluator is easily transformed to a proof usQ_{mid} for the usurper. This attack is easily countered by having all evaluators publish a zero-knowledge proof of knowledge of their secret exponent s, along with their public key $s\phi_1(P)$.

6 Challenges in implementing isogeny-based delay functions

For a delay function to be useful, there need to be convincing arguments as to why the evaluation cannot be performed considerably better than with the legitimate algorithm.

In this sense, repeated squaring modulo an RSA modulus is especially appealing: modular arithmetic has been studied for a long time, and we are reasonably confident that we know all useful algorithms and hardware in this respect; and the repeated application of the function $x \mapsto x^2$ is so simple that one may hope no better algorithm exists (see [1], though).

Repeated squaring in class groups, already, raises more skepticism, as the arithmetic of class groups is a much less studied area. This clearly had an impact on Ethereum's choice to go with RSA-based VDFs, despite class group based ones not needing a trusted setup.

For isogeny based delay functions, we argue that the degree of assurance seems to be nearly as good as for RSA based ones, although more research is certainly needed. To support this claim, we give here more details on the way the evaluation of $\hat{\phi}$ is performed, that were omitted by [17].

For a start, we must choose a prime degree ℓ. Intuitively, the smaller, the better, thus we shall fix $\ell = 2$, although $\ell = 3$ also deserves to be studied. A 2-isogeny is represented by rational maps of degree 2, thus we expect one isogeny evaluation to require at least one multiplication modulo p. Our goal is to get as close as possible to this lower bound, by choosing the best representation for the elliptic curves, their points, and their isogenies.

It is customary in isogeny based cryptography to use curves in Montgomery form, and projective points in $(X : Z)$ coordinates, as these give the best formulas for arithmetic operations and isogenies [13,28]. Montgomery curves satisfy the equation

$$E \ : \ y^2 = x^3 + Ax^2 + x,$$

in particular they have a point of order two in $(0, 0)$, and two other points of order two with x-coordinates α and $1/\alpha$, where α is a root of the polynomial x^2+Ax+1, and possibly lives in $\mathbb{F}_{p^2}$. These three points define the three possible isogenies of degree 2 starting from E. The Montgomery form is almost unique, there being only six possible choices for the A coefficient for a given isomorphism class, corresponding to the three possible choices for the point to send in $(0, 0)$ (each taken twice).

In our case, all three points (in projective coordinates) $(0 : 1)$, $(\alpha : 1)$ and $(1 : \alpha)$, are defined over $\mathbb{F}_p$, we thus choose to distinguish one additional point by writing the curves as

$$E_\alpha \ : \ y^2 = x(x - \alpha)(x - 1/\alpha),$$

with $\alpha \neq 0, \pm 1$. We call this a *semi-Montgomery form*; although it is technically equivalent to the Montgomery form, 2-isogeny formulas are expressed in it more easily. Recovering the Montgomery form is easy via $A = -\alpha - 1/\alpha$.

Using the formula of Renes [28], we readily get the isogeny with kernel generated by $(\alpha : 1)$ as

$$\phi_\alpha(x, y) = \left(x\frac{x\alpha - 1}{x - \alpha}, \dots \right), \tag{2}$$

and its image curve is the Montgomery curve defined by $A = 2 - 4\alpha^2$. By comparing with the multiplication-by-2 map on E_α, we obtain the dual map to ϕ_α as

$$\hat{\phi}_\alpha(x, y) = \left(\frac{(x+1)^2}{4\alpha x}, \dots \right). \tag{3}$$

It is clear from this formula that the kernel of $\hat{\phi}_\alpha$ is generated by $(0, 0)$.

This formula is especially interesting, as we verify that its projective version in $(X : Z)$ coordinates only requires 2 multiplications and 1 squaring:

$$\hat{\phi}_\alpha(X : Z) = \big((X + Z)^2 : 4\alpha XZ\big), \tag{4}$$

and the squaring can be performed in parallel with one multiplication. The analogous formulas for $\phi_{1/\alpha}$ are readily obtained by replacing $\alpha \to 1/\alpha$ in the previous ones, and moving around projective coefficients to minimize work.

But, if we want to chain 2-isogenies, we need a way to compute the semi-Montgomery form of the image curve. For the given $A = 4\alpha^2 - 2$, direct calculation shows that the two possible choices are

$$\alpha' = 2\alpha\left(\alpha \pm \sqrt{\alpha^2 - 1}\right) - 1 = \left(\alpha \pm \sqrt{\alpha^2 - 1})\right)^2. \tag{5}$$

As we know that $(0, 0)$ generates the dual isogeny to ϕ_α, neither choice of α' will define a backtracking walk. Interestingly, Castryck and Decru [10] show that when $p = 7 \mod 8$, if $\alpha \in \mathbb{F}_p$, if ϕ_α is a horizontal isogeny (see definition in [10]), and if α' is defined as

$$\alpha' = \left(\alpha + \sqrt{\alpha^2 - 1})\right)^2$$

where $\sqrt{\alpha^2 - 1}$ denotes the principal square root, then $\alpha' \in \mathbb{F}_p$ and $\phi_{\alpha'}$ is horizontal too. This gives a very simple algorithm to perform a non-backtracking 2-isogeny walk staying in the $\mathbb{F}_p$-restricted isogeny graph, i.e., a walk on the 2-*crater*. Alternatively, if a pseudo-random walk in the full supersingular graph is wanted, one simply takes a random square root of $\alpha^2 - 1$.

Using these formulas, the isogeny walk $\phi : E \to E'$ is simply represented by the list of coefficients α encountered, and the evaluation of $\hat{\phi}$ using Formula (4) costs 2 multiplications and 1 parallel squaring per isogeny.

Implementation challenges. Following the recommendations of [17], for a 128-bits security level we need to choose a prime p of around 1500 bits, which is comparable to the 2048-bits RSA arithmetic targeted by Ethereum, although possibly open to optimizations for special primes.

In software, the latency of multiplication modulo such a prime is today around 1μs. The winner of the Ethereum FPGA competition [33] achieved a

latency of 25ns for 2048-bits RSA arithmetic. Assuming a pessimistic baseline of 50ns for one 2-isogeny evaluation, for a target delay of 1 hour we need an isogeny walk of length $\approx 7 \cdot 10^{10}$. That represents as many coefficients α to store, each occupying ≈ 1500 bits, i.e., ≈ 12TiB of storage!

We stress that only the evaluation key ek requires such large storage, and thus only evaluators (running extraction in Delay Encryption, or evaluating a VDF) need to store it. However any FPGA or hardware design for the evaluation of isogeny-based delay functions must take this constraint into account, and provide fast storage with throughputs of the order of several GiB/s.

At present, we do not know any configuration that pushes these 2-isogeny computations into being memory bandwidth bound. In fact, computational adversaries only begin encountering current DRAM and CPU bus limits when going an order of magnitude faster than the hypothetical high speeds above.

An isogeny-based VDF could dramatically reduce storage requirements by doing repeated shorter evaluations, and simply hashing each output to be the input for the next evaluation. We sacrifice verifier time by doing so, but verifiers remain fast since they still only compute two pairings per evaluation. We caution however that this trick does not apply to Delay Encryption.

In [17], De Feo *et al.* describe an alternative implementation that divides the required storage by a factor of 1244, at the cost of slowing down evaluation by a factor of at least $\log_2(1244)$. While this trade-off could be acceptable in some non-fully distributed applications, it seems incompatible with applications where evaluators want to get to the result as quickly as possible, e.g., when several evaluators are competing to compute the output.

It would be very interesting to find compact representations of very long isogeny chains which do not come at the expense of efficiently evaluating them.

Optimality. Formula (4) is, intuitively, almost optimal, as we expect that a 2-isogeny in projective $(X : Z)$ coordinates should require at least 2 multiplications. And indeed we know of at least one case where a 2-isogeny can be evaluated with 2 parallel multiplications: the isogeny of kernel $(0 : 1)$ is given by

$$\phi_0(x, y) = \left(\frac{(x-1)^2}{x}, \dots \right), \tag{6}$$

or, in projective coordinates,

$$\phi_0(X : Z) = \big((X - Z)^2 : XZ\big), \tag{7}$$

which only requires one parallel multiplication and squaring.

We tried to construct elliptic curve models and isogeny formulas that could evaluate 2-isogeny chains using only 2 parallel multiplications per step, however any formula we could find had a coefficient similar to α intervene in it, and thus bring the cost up by at least one multiplication.

Intuitively, this is expected: there are exponentially many isogeny walks, and the coefficients α must necessarily intervene in the formulas to distinguish

between them. However this is far from being a proof. Even proving a lower bound of 2 *parallel* multiplications seems hard.

It would be interesting to prove that any 2-isogeny chain needs at least 2 *sequential* multiplications for evaluation, or alternatively find a better way to represent and evaluate isogeny chains.

7 Conclusion

We introduced a new time delay primitive, named Delay Encryption, related to Time-lock Puzzles and Verifiable Delay Functions. Delay Encryption has some interesting applications such as sealed-bid auctions and electronic voting. We gave an instantiation of Delay Encryption using isogenies of supersingular curves and pairings, and discussed several related topics that also apply to the VDF of De Feo, Masson, Petit and Sanso.

Several interesting questions are raised by our work, such as, for example, clarifying the relationship between Delay Encryption, Verifiable Delay Functions and Time-lock Puzzles.

Like the isogeny-based VDF, our Delay Encryption requires a trusted setup. We described an efficient way to perform a distributed trusted setup, however the associated zero-knowledge property relies on a non-falsifiable assumption which requires more scrutiny.

The implementation of delay functions from isogenies presents several practical challenges, such as needing very large storage for the public parameters. On top of that, it is not evident how to prove the optimality of isogeny formulas used for evaluating the delay function. While we gave here extremely efficient formulas, these seem to be at least one multiplication more expensive than the theoretical optimum. More research on the arithmetic of elliptic curves best adapted to work with extremely long chains of isogenies is needed.

Finally, we invite the community to look for more constructions of Delay Encryption, in particular quantum-resistant ones.

References

1. Bernstein, D.J., Sorenson, J.: Modular exponentiation via the explicit Chinese remainder theorem. Mathematics of Computation **76**, 443–454 (2007). https://doi.org/10.1090/S0025-5718-06-01849-7
2. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 227–247. Springer International Publishing, Cham (2019)
3. Bitansky, N., Goldwasser, S., Jain, A., Paneth, O., Vaikuntanathan, V., Waters, B.: Time-lock puzzles from randomized encodings. In: Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science. p. 345–356. ITCS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2840728.2840745

4. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 757–788. Springer International Publishing, Cham (2018)
5. Boneh, D., Franklin, M.: Efficient generation of shared RSA keys. In: Kaliski, B.S. (ed.) Advances in Cryptology — CRYPTO '97. pp. 425–439. Springer Berlin Heidelberg, Berlin, Heidelberg (1997)
6. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. SIAM Journal on Computing **32**(3), 586–615 (2003). https://doi.org/10.1137/S0097539701398521
7. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology **17**(4), 297–319 (Sep 2004). https://doi.org/10.1007/s00145-004-0314-9
8. Bowe, S., Chiesa, A., Green, M., Miers, I., Mishra, P., Wu, H.: Zexe: Enabling decentralized private computation. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 947–964 (2020). https://doi.org/10.1109/SP40000.2020.00050
9. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In: Hofheinz, D., Rosen, A. (eds.) Theory of Cryptography. pp. 407–437. Springer International Publishing, Cham (2019)
10. Castryck, W., Decru, T.: CSIDH on the surface. In: Ding, J., Tillich, J.P. (eds.) Post-Quantum Cryptography. pp. 111–129. Springer International Publishing, Cham (2020)
11. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S.D. (eds.) Advances in Cryptology – ASIACRYPT 2018. pp. 395–427. Springer International Publishing (2018)
12. Castryck, W., Panny, L., Vercauteren, F.: Rational isogenies from irrational endomorphisms. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020. pp. 523–548. Springer International Publishing, Cham (2020)
13. Costello, C., Longa, P., Naehrig, M.: Efficient algorithms for Supersingular Isogeny Diffie-Hellman. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016: 36th Annual International Cryptology Conference. pp. 572–601. Springer Berlin Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_21
14. De Feo, L., Galbraith, S.D.: SeaSign: Compact isogeny signatures from class group actions. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019. pp. 759–789. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_26
15. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology **8**(3), 209–247 (2014)
16. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: Compact post-quantum signatures from quaternions and isogenies. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 64–93. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64837-4_3
17. De Feo, L., Masson, S., Petit, C., Sanso, A.: Verifiable delay functions from supersingular isogenies and pairings. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 248–277. Springer International Publishing, Cham (2019)

18. Eisenträger, K., Hallgren, S., Lauter, K., Morrison, T., Petit, C.: Supersingular isogeny graphs and endomorphism rings: Reductions and solutions. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 329–368. Springer International Publishing (2018)
19. Gabizon, A., Williamson, Z.J.: plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Report 2020/315 (2020), https://eprint.iacr.org/2020/315
20. Galbraith, S.D., Vercauteren, F.: Computational problems in supersingular elliptic curve isogenies. Quantum Information Processing **17**(10), 265 (2018)
21. Howard, M., Cohen, B.: Chia network announces 2nd VDF competition with $100,000 in total prize money (2019), https://www.chia.net/2019/04/04/chia-network-announces-second-vdf-competition-with-in-total-prize-money.en.html
22. Kohel, D.R., Lauter, K., Petit, C., Tignol, J.P.: On the quaternion-isogeny path problem. LMS Journal of Computation and Mathematics **17**(A), 418–432 (2014)
23. Kutas, P., Martindale, C., Panny, L., Petit, C., Stange, K.E.: Weak instances of SIDH variants under improved torsion-point attacks. Cryptology ePrint Archive, Report 2020/633 (2020), https://eprint.iacr.org/2020/633
24. Love, J., Boneh, D.: Supersingular curves with small noninteger endomorphisms. Open Book Series **4**(1), 7–22 (2020)
25. Malavolta, G., Thyagarajan, S.A.K.: Homomorphic time-lock puzzles and applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 620–649. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26948-7_22
26. Petit, C.: Faster algorithms for isogeny problems using torsion point images. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. pp. 330–353. Springer International Publishing (2017)
27. Pietrzak, K.: Simple Verifiable Delay Functions. In: Blum, A. (ed.) 10th Innovations in Theoretical Computer Science Conference (ITCS 2019). Leibniz International Proceedings in Informatics (LIPIcs), vol. 124, pp. 60:1–60:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018). https://doi.org/10.4230/LIPIcs.ITCS.2019.60
28. Renes, J.: Computing isogenies between Montgomery curves using the action of (0, 0). In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography. pp. 229–247. Springer International Publishing (2018)
29. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Tech. rep., Cambridge, MA, USA (1996), https://people.csail.mit.edu/rivest/pubs/RSW96.pdf
30. Delpech de Saint Guilhem, C., Kutas, P., Petit, C., Silva, J.: SÉTA: supersingular encryption from torsion attacks. Cryptology ePrint Archive, Report 2019/1291 (2019), https://eprint.iacr.org/2019/1291
31. Shlomovits, O.: Diogenes Octopus: Playing red team for Eth2.0 VDF (Jun 2020), https://medium.com/zengo/dac3f2e3cc7b
32. Shlomovits, O.: Dogbyte attack: Playing red team for Eth2.0 VDF (Aug 2020), https://medium.com/zengo/ea2b9b2152af
33. VDF Alliance: VDF FPGA competition (2019), https://supranational.atlassian.net/wiki/spaces/VA/pages/36569208/FPGA+Competition
34. Wesolowski, B.: Efficient verifiable delay functions. In: EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 379–407. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17659-4_13