A note on isogeny-based hybrid verifiable delay functions

Barak Shani
University of Pennsylvania

Abstract

Using the idea behind the recently proposed isogeny- and paring-based verifiable delay function (VDF) by De Feo, Masson, Petit and Sanso, we construct an isogeny-based VDF without the use of pairings. Our scheme is a hybrid of time-lock puzzles and (trapdoor) verifiable delay functions. We explain how to realise the proposed VDF on elliptic curves with commutative endomorphism ring, however this construction is not quantum secure. The more interesting, and potentially quantum-secure, non-commutative case is left open.

1 Introduction

The idea of using cryptography to force an entity to invest a desired amount of time in some task was first proposed by Timothy C. May [14], and put forward by Rivest, Shamir and Wagner [17]. Aiming at enabling others to verify that the amount of time was indeed spent on completing the task, but without re-doing the task, Boneh, Bonneau, Bünz and Fisch [3] proposed the notion of a "verifiable delay function" (VDF), which formalises some of the aspects in the earlier work of Lenstra and Wesolowski [13]. Informally, a VDF is a function $f : X \to Y$ that takes a prescribed time to compute, yet the correctness of an input–output pair (x, y) can be quickly verified. Of particular importance is the *sequentiality* in the function evaluation, that prevents any (substantial) advantage by parallel computations.

Wesolowski [22] and Pietrzak [16] independently proposed similar constructions that are based on the sequentiality of exponentiation in the group $\mathbb{Z}_N^*$ of an unknown order (e.g. for an RSA modulus N). These constructions are reviewed in [2], where the problem of construction a quantum-secure VDF is left open (due to Shor's quantum algorithm [19], the proposed constructions are not quantum secure). To that end, De Feo, Masson, Petit and Sanso [10] explored the possibility of constructing a VDF from elliptic curve isogenies. Their isogeny-based VDF relies on the sequentiality in the isogeny evaluation (the need to visit all nodes in a given path on the isogeny graph), however their verification algorithm is based on elliptic curve pairings, and thus the construction is not quantum secure (since breaking the pairings allows one to easily compute elements that pass the verification test).

In this short note we build upon the previous construction and propose an isogeny-based *trapdoor* VDF without the use of pairings. A trapdoor VDF is equipped with a trapdoor that allows evaluating

the function efficiently, see [22]. The main idea behind this construction is to perform an *unbalanced* key-exchange, where the evaluating party takes a long path on the isogeny graph while the verifying party follows a short path. We consider a static-ephemeral key exchange, where the static key is public.

Our construction can be realised over elliptic curves with commutative endomorphism ring. Unfortunately, in this case our proposed construction is quantum insecure. A quantum attack is not known for the non-commutative supersingular case, however we do not offer a practical realisation of our construction in this case. We present it as an open problem,[1] which is the main aim in releasing this note.

2 Verifiable delay functions

A verifiable delay function, as defined in [2], consists of three algorithms:

- $Setup(\lambda, T) \to pp$: an initial setup phase that takes a security parameter λ and a time bound T and outputs public parameters pp.
- $Eval(pp, x) \to (y, \pi)$: an evaluation procedure that takes $x \in X$ and outputs $y \in Y$ and a proof π.
- $Verify(pp, x, y, \pi) \to \{accpet, reject\}$: a verification procedure that outputs $accept$ if y is the correct evaluation of the VDF on x.

A detailed definition is given [3, Section 3], along with the requirement of *correctness* (verification accepts on valid input–output pairs (x, y)), *soundness* (verification rejects on invalid pairs) and sequentiality, mentioned above. A VDF may require an interactive engagement between the different parties, either in the choice of the input x or in the verification procedure.

2.1 A VDF–time-lock-puzzle hybrid

We propose a construction that can be seen as a hybrid of (trapdoor) verifiable delay functions and time-lock puzzles (formal definitions can be found in [1, Section 3.1]) and so we call it *hybrid VDF*[2]. The protocol is interactive, where we explicitly add a *challenge* phase, in which a challenger chooses an input $x \in X$, and obtains a hint (trapdoor) h that can be used by anyone to efficiently verify the correct output of the function.[3]

The proposed construction deviates from the definition of VDF in the following ways:

[1] Thereby increasing the pool of open problems in isogeny-based cryptography [4].

[2] An alternative proper name is *verifiable time-lock puzzles*, however we believe that the VDF construction is better suited.

[3] We do not discuss concrete ways to generate the challenge. A verifiable method may be desired, as well as the ability to re-generate the trapdoor (so it could be broadcasted) in case it kept secret [5].

- The setup phase takes time linear in T, and thus may not be polynomial in the security parameter λ (when T is super-polynomial in λ).

- In order to publicly verify the work, the hint should be broadcasted (after (y, π) is received).

In accordance with [1, Section 4], the first point may be thought of as one-time *pre-processing* phase. The second point may resemble similar issues for time-lock puzzles. However, verification is not an essential feature of time-lock puzzles – a trapdoor, if exists, is not even supposed to be remembered. Secondly, in some proposals, once the puzzle has been solved, there is no more use in the public parameters, and therefore one has to complete a new setup in order to create a new puzzle. In our construction, revealing the trapdoor does not result in abandoning the parameters from the setup phase. Hence, we use the trapdoor differently from the trapdoor VDF in [22].

3 A hybrid verifiable delay function

The following does not require deep understanding of isogeny-based cryptography, however familiarity with the terminology may be helpful. Some facts can be found in [10, Section 4], and a good introduction to isogeny-based key exchange can be found in the survey by Smith [20].

Consider ordinary elliptic curves (as in Couveignes–Rostovtsev–Stolbunov key exchange [7, 18]) or supersingular that are defined over the prime field $\mathbb{F}_p$ (as in CSIDH [6]). The ideal class group $\mathrm{cl}(\mathcal{O})$, which acts via isogenies on isomorphism classes of such curves, is commutative (here $\mathcal{O}$ is an order in an imaginary quadratic field).

A Diffie–Hellman-style key exchange proceeds as follows: Alice and Bob select privately $[\mathfrak{a}], [\mathfrak{b}] \in \mathrm{cl}(\mathcal{O})$, respectively; they publish the curves $[\mathfrak{a}]E$ and $[\mathfrak{b}]E$, respectively, for some agreed curve E; acting on the other party's curve by their own private key, both obtain the curve $[\mathfrak{a}][\mathfrak{b}]E = [\mathfrak{b}][\mathfrak{a}]E$. The shared key can be taken to be the j-invariant of the shared curve or a coefficient in a canonical representation.

Essentially, our construction revolves around the sequentiality of the function $f : X \to Y$, where $X = Y$ is a set of representatives of elliptic curve isomorphism classes (e.g. j-invariants), and f is an isogeny between elliptic curves, which for convenience we represent as a path on the isogeny graph. We say that an isogeny is T-*long* if it takes at least time T to compute.

A hybrid trapdoor VDF can be obtained by setting up an elliptic curve E, a T-long isogeny $[\mathfrak{a}]$ and the curve $[\mathfrak{a}]E$. The challenger selects an isogeny $[\mathfrak{b}]$ and publishes $[\mathfrak{b}]E$. The evaluation task is to compute $[\mathfrak{a}]([\mathfrak{b}]E)$, which can be verified easily by computing $[\mathfrak{b}]([\mathfrak{a}]E)$. Note that only the challenger possess $[\mathfrak{b}]$, so once the task is declared completed, the challenger publishes $[\mathfrak{b}]$ (either before or only after verifying the validity of the work).

Formally, we have

- $Setup(\lambda, T)$ outputs the public parameters pp: a prime p, an elliptic curve $E/\mathbb{F}_p$, an ideal class $[\mathfrak{a}]$ for a T-long isogeny, and the curve $[\mathfrak{a}]E$.
- $Challenge(pp, T)$ produces a private ideal class $[\mathfrak{b}]$ and its representation h, and the curve $x := [\mathfrak{b}]E$.[4]
- $Eval(pp, x)$ computes $[\mathfrak{a}]([\mathfrak{b}]E)$ and outputs the shared key y and an empty proof π.
- $Verify(pp, x, y; h)$ computes, using the hint, $[\mathfrak{b}]([\mathfrak{a}]E)$ and accepts if the shared key is y; otherwise it rejects.

Correctness follows from the correctness of the key exchange procedure, and soundness holds because the verification procedure only accepts on the correct value.

Realisation: The evaluation and verification procedures are clear, as they follow an already established key exchange protocol. As we now explain, representation of the isogenies in the setup and challenge phases also follows previous work. In CSIDH [6],

- We set $p = 4\ell_1 \cdots \ell_n - 1$ for distinct small odd primes ℓ_j, the corresponding ideals $[\mathfrak{l}_j]$ and a supersingular curve E over $\mathbb{F}_p$.
- Set $[\mathfrak{a}] = [\mathfrak{l}_1^{e_1} \dots \mathfrak{l}_n^{e_n}]$, for some values e_j such that the isogeny computation, corresponding to $[\mathfrak{a}]$, is T-long (for example, [10] takes a path on a single cycle corresponds to a single ℓ_j).
- Set $[\mathfrak{b}] = [\mathfrak{l}_1^{e'_1} \cdots \mathfrak{l}_n^{e'_n}]$ for e'_j randomly selected from some range $\{-m, \dots, m\}$, computed such that recovering all e'_j from $[\mathfrak{b}]E$ takes time at least T, and $h = (e'_1, \dots, e'_n)$.

The case of ordinary elliptic curves is similar with small adjustments, see [9].

Security and sequentiality: since we use a similar evaluation function to [10], many of the security arguments follow (see Section 6 there). Moreover, as also stated in the previous work, among the known algorithms for isogeny computation, the most efficient ones require to visit all intermediate curves in the isogeny path; working in parallel brings very little advantage for this task.

Of particular importance is the hardness of computing $[\mathfrak{a}]E$ in shorter time than T, which is possible in general (i.e. a smaller-degree isogeny exists). This requires to find a shorter path on the isogeny graph – a problem that is considered to be hard on classical computers. However, there is a difference between the ordinary and supersingular cases. Current methods to generate supersingular curves start with special curves that make the isogeny path problem easy; thus in [10] the setup is required to run by a trusted authority that does not reveal how the supersingular curve was constructed; see full details

[4] The time parameter T is an input to $Challenge$ since the generation of $[\mathfrak{b}]$ should guarantee that computing $[\mathfrak{b}]$ from E and $[\mathfrak{b}]E$ takes time at least T.

in [10, Section 6.2]. Ordinary curves seem to avoid this requirement, and in our construction they do not suffer from the insecure instantiation mentioned at the end of [10, Section 6.2].

Note that computing $[\mathfrak{b}]$ from E and $[\mathfrak{b}]E$ is the "isogeny problem", and that computing the shared key without following the protocol is the "computational isogeny Diffie–Hellman problem". Both these problems are considered quantumly hard (and in fact are quantumly equivalent [11]).

Quantum computers would break the security of our function, as explained in the previous work: using a variation of Shor's algorithm, one can efficiently compute a shorter path on the isogeny graph between E and $[\mathfrak{a}]E$ (see also [8, Appendix C]). We remark that in our construction a shorter path can be computed classically in subexponential time, see [9, Section 5.2] or [8, Appendix C] (thus T needs to be smaller than the running time of this algorithm).

Remark 1. Our proposed construction can be turned into an *incremental* one, i.e. a single setup supports multiple hardness parameters T (see [3]). As remarked in [10], this can be done by including in the public parameters several pit stops along the isogeny path of $[\mathfrak{a}]$, marking a specific difficulty level. This is highly desired in our construction, since the setup phase may take a long time to complete. Note that the commutativity allows us to compute $[\mathfrak{a}]$ in several different ways (unless it is a path on a single cycle), thus in this case the order in which $[\mathfrak{a}]$ is computed should be agreed on.

3.1 The non-commutative case – an open problem

SIDH [12] seems to naturally fit our construction, since there is an inherent lack of symmetry between the two parties (see for example [20, Section 15]). Interestingly, Petit [15] already suggested to use unbalanced SIDH for the static-ephemeral setting, similarly to our proposal, for the reason of protecting the static key more strongly. He provides several attacks on unbalanced SIDH on special curves.

In SIDH, Alice defines her isogeny using the torsion group $E[N_a]$ (Bob works over a different group $E[N_b]$), for some prime power N_a. Essentially, she selects a point in this group and sets the subgroup it generates to be the kernel of her (unique) isogeny. Moreover, by generating the isogeny in this way, Bob is able to compute the shared curve later in the protocol. However, the number of steps in the isogeny path coming from this procedure is at most $\log N_a$. Since the static path in our construction should be very long, it means that N_a should be very large. This directly affects the size of the base field, hence not suitable for our needs.

Alternatively, one could take a prime N_a, where there are no intermediate steps along the path and the amount of computations to complete the isogeny evaluation is proportional to $O(N_a)$. The downside of this approach is that this isogeny computation (using Vélu's formulas [21]) do enjoy parallelisation, hence also not suitable for our needs. Lastly, the static key could in principle be created by a long random walk, but we are not aware of any method that allows to complete a key share this way.

We leave this interesting case as an open problem.

Acknowledgements

Thanks to Antonio Sanso for sharing, together with Simon Masson and Luca De Feo, an early draft of their work, and to Luca De Feo for helpful discussions. This work was supported by the National Science Foundation under grant no. CNS-1513671 and the ONR SynCrypt project.

References

[1] Bitansky, N., Goldwasser, S., Jain, A., Paneth, O., Vaikuntanathan, V., and Waters, B. (2016) "Time-Lock Puzzles from Randomized Encodings," in ACM Conference on Innovations in Theoretical Computer Science – ITCS '16, pp. 345–356. ACM, New York, NY.

[2] Boneh, D., Bünz, B., and Fisch, B. (2018) "A Survey of Two Verifiable Delay Functions," in Cryptology ePrint Archive, Report 2018/712. http://eprint.iacr.org/2018/712.

[3] Boneh, D., Bonneau, J., Bünz, B., and Fisch, B. (2018) "Verifiable Delay Functions," in Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. LNCS, vol. 10991, pp. 757–788. Springer, Heidelberg.

[4] Boneh, D., Glass, D., Krashen, D., Lauter, K., Sharif, S., Silverberg, A., Tibouchi, M., and Zhandry, M. (2018) "Multiparty Non-Interactive Key Exchange and More From Isogenies on Elliptic Curves," in MathCrypt 2018 (in print).

[5] Boneh, D., and Naor, M. (2000) "Timed Commitments," in Bellare, M. (ed.) Advances in Cryptology — CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Berlin, Heidelberg.

[6] Castryck, W., Lange, T., Martindale, C., Panny, L., and Renes, J. (2018) "CSIDH: An Efficient Post-Quantum Commutative Group Action," in Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology – ASIACRYPT 2018. LNCS, vol. 11273, pp. 395–427. Springer, Heidelberg.

[7] Couveignes, J.-M. (2006) "Hard Homogeneous Spaces," in Cryptology ePrint Archive, Report 2006/291. http://eprint.iacr.org/2006/291.

[8] De Feo, L., and Galbraith S.D. (2018) "SeaSign: Compact Isogeny Signatures from Class Group Actions," in Cryptology ePrint Archive, Report 2018/824. http://eprint.iacr.org/2018/824.

[9] De Feo, L., Kieffer, J., and Smith, B. (2018) "Towards practical key exchange from ordinary isogeny graphs," in Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology – ASIACRYPT 2018. LNCS, vol. 11273, pp. 365–394. Springer, Heidelberg.

[10] De Feo, L., Masson, Petit, C., S., and Sanso, A. (2019) "Verifiable Delay Functions from supersingular isogenies and pairings," in Cryptology ePrint Archive, Report 2019/166. http://eprint.iacr.org/2019/166.

[11] Galbraith, S., Panny, L., Smith, B., and Vercauteren, F. (2018) "Quantum Equivalence of the DLP and CDHP for Group Actions," in Cryptology ePrint Archive, Report 2018/1199. http://eprint.iacr.org/2018/1199.

[12] Jao, D., and De Feo, L. (2011) "Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies," in Yang, B.-Y. (ed.) Post-Quantum Cryptography – PQCrypto 2011. LNCS, vol. 7071, pp. 19–34. Springer, Heidelberg.

[13] Lenstra, A.K., and Wesolowski, B. (2017) "Trustworthy Public Randomness with Sloth, Unicorn, and Trx," in International Journal of Applied Cryptography, **3**(4), 330–343.

[14] May, T.C. (1993) "Timed-Release Crypto," message on the cypherpunks mailing list. http://cypherpunks.venona.com/date/1993/02/msg00129.html.

[15] Petit, C. (2017) "Faster Algorithms for Isogeny Problems Using Torsion Point Images," in Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. LNCS, vol. 10625, pp. 330–353. Springer, Heidelberg.

[16] Pietrzak, K. (2018) "Simple Verifiable Delay Functions," in Cryptology ePrint Archive, Report 2018/627. http://eprint.iacr.org/2018/627.

[17] Rivest, R.L., Shamir, A. and Wagner, D.A. (1996) "Time-lock Puzzles and Timed-release Crypto," Technical Report.

[18] Rostovtsev, A., and Stolbunov, A. (2006) "Public-Key Cryptosystem Based on Isogenies," in Cryptology ePrint Archive, Report 2006/145. http://eprint.iacr.org/2006/145.

[19] Shor, P.W. (1997) "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," in SIAM Journal on Computing, **26**(5), 1484–1509.

[20] Smith, B. (2018) "Pre- and Post-quantum Diffie–Hellman from Groups, Actions, and Isogenies," in Budaghyan L., Rodríguez-Henríquez F. (eds) Arithmetic of Finite Fields, WAIFI 2018. LNCS, vol. 11321, pp. 3–40. Springer, Cham.

[21] Vélu, J. (1971) "Isogénies entre courbes elliptiques," in Comptes Rendus de l'Académie des Sciences de Paris, Série A, **273**, 238–241.

[22] Wesolowski, B. (2018) "Efficient Verifiable Delay Functions," in Cryptology ePrint Archive, Report 2018/623. http://eprint.iacr.org/2018/623. (to appear at Eurocrypt 2019)