Construct a Verifiable delay function (VDF) using a trapdoor where the VDF can be efficiently evaluated by parties that know the secret trapdoor. In this construction, the trapdoor is unknown even to the party running the setup environment so this does not require any trusted setup to evaluate. Here, the construction is based on RSA group(unknown order).

Implementation Details

(a) **VDF**: Verifiable delay function is slow to compute and easy to verify. This is computed in a prescribed amount of time, t and cannot be performed faster even if it is parallelizable.

(b) **The Protocol (Wesolowski Prover)**: VDF construction consists of solving an instance of the RSW time-lock puzzle which takes the input as RSA group and a timing parameter, t and compute $y = x^{2^t}$

(c) **Trapdoor verifiable delay function**: The construction proposed in the paper is a trapdoor VDF from which a VDF is derived. The prover, P holds the secret key, sk (the trapdoor), and an associated public key, pk. Given x, a trapdoor VDF allows to compute output y from x and anyone can verify if y is computed by the prover or the computation of y required an amount of time at least δ

- **Keygen** $\rightarrow(pk, sk)$ is a key generation procedure, which outputs Provers's public key pk and secret key sk and the public key is be publicly available.
- **Trapdoor**$_{sk}(x,\delta)\rightarrow(y,\pi)$ takes as input the data x $\in \mathbb{X}$ (for some input space X) and uses the secret key sk to produce the output y from x. The parameter δ is the amount of sequential work required to compute the same output y without knowledge of the secret key.
- **Evaluation**$_{pk}(x,\delta)\rightarrow(y,\pi)$ is a procedure to evaluate the function on x using only the public key pk, for a targeted amount of sequential work δ. It produces the output y from x, and proof of π.
- **Verify**$_{pk}(x, y,\pi,\delta) \rightarrow$ **true** or **false**. Here, we check if y is indeed the correct output for x, associated to the public key pk and the evaluation time δ, possibly with the help of the proof π.

References

(a) Verifiable Delay Functions. Boneh, 2018 [Link]

(b) Efficient verifiable delay functions. Wesolowski, 2018 [Link]

(c) Simple Verifiable Delay Functions. Pietrzak, 2018 [Link]

(d) Time-lock puzzles and timed-release Crypto, [Link]