I listened to the talk of Continuous Verifiable Delay Functions (cVDF,https://eprint.iacr.org/2019/619.pdf) at the virtual Eurocrypt yesterday, and came up with this idea. All kinds of feedback are welcome.

In traditional VDF, only the last output is verifiable in a polylog time. In cVDF, the output of each iteration is verifiable in a polylog time.

We can use this property to construct non-parallelisable PoW mining. In particular, we replace the hash function in PoW mining with cVDF. Given the metadata (previous hash, merkle root, …), a miner calculates the cVDF output of each iteration, until finding an output that satisfies the difficulty requirement.

VDF and cVDF

VDF is defined as follows (copied fromhttps://eprint.iacr.org/2018/712.pdf).

- \mathsf{Setup}(\lambda,T) \to pp

: Probabilistic. On input a security parameter \lambda

and a time bound T

, outputs public parameters pp

.

- \mathsf{Eval}(pp,x)→(y,\pi)

: Deterministic. On input pp

and a x \in X

, outputs a y \in Y

and a proof \pi

.

- \mathsf{Verify}(pp, x, y, \pi) → {0, 1}

: Deterministic. Outputs 1

if y

is the correct evaluation of the VDF on input x

.

Here \mathsf{Eval}(\cdot)

is an iterative process. There is an intermediate value, which is updated for each iteration. Using multiple cores/computers cannot accelerate its computation.

In this VDF, only the final output y

is verifiable. In cVDF, each of the intermediate value is also verifiable. Unfortunately the cVDF paper does not list the syntax of it, so we informally define it here (might be inaccurate).

- Let y_1, y_2, \dots, y_n

be the first to the last output.

- Let \mathsf{Eval}(pp,x, y_t, \pi_t)→(y_{t+1},\pi_{t+1})

be the function that evaluates value y_t

and outputs the next value y_{t+1}

and its proof \pi_{t+1}

.

- Let \mathsf{Verify}(pp, t, x, y_t, \pi_t) → {0, 1}

be the function that verifies the output y_t

.

Non-parallelisable PoW mining

Mining is parallelisable: the miner can enumerate a large amount of nonces and compute them in parallel. This enables miners to accelerate the mining by having lots of mining hardware.

We construct non-parallelisable mining from cVDF. We replace hash functions of PoW with cVDFs. Rather than using H(prev_hash || merkle_root || time || nBits || nonce)

, the miner computes as follows.

- pp \gets \mathsf{Setup}(\lambda,T)

where T

is extremely big

- x \gets prev_hash || merkle_root || time || nBits

(note that there is no nonce anymore)

- y_0, \pi_0 \gets \mathsf{cVDF}(pp, x)
- t = 0
- While t++
- y_t, \pi_t \gets \mathsf{cVDF}(pp, x, y_{t-1}, \pi_{t-1})
- If y_t < Target

, the miner finds the block.

- y_t, \pi_t \gets \mathsf{cVDF}(pp, x, y_{t-1}, \pi_{t-1})
- If y_t < Target

, the miner finds the block.

In this way, mining is non-parallelisable.

Combining with non-outsourceable mining

Recall that we introduced VRF-based mining before[Preventing pooled mining by VRF-based mining]. It simply replaces hash using VRF. By doing this, the pool operator should send his secret key in order to outsource mining to other miners. In this way, mining is no longer outsourceable.

We can combine VRF-based mining with this cVDF based mining easily. Instead of comparing y_t

with Target

directly, the miner calculates y'_t \gets \mathsf{VRFHash}(y_t, ...)

, and compares y'_t

with Target

.

In this way, mining is non-outsourceable and non-parallelisable. This is ultimately egalitarianism.

Known issues

Miners may still have opportunity to parallelise mining, by tweaking the merkle root. We consider the following scenarios: enumerating key pairs, changing notes of txs, selecting different sets of transactions.

Enumerating secret keys/public keys

One can enumerate lots of sk/pk, so have lots of coinbase txs and can compute cVDF as many times as he can. The countermeasure I can think of is to enforce a stake requirement to sk/pk.

In particular, Target of each sk/pk is a function of his contribution to this blockchain, i.e., number of blocks he mined in the last x blocks. With more contribution, mining a block is easier. By tweaking this function, parallising-by-enumerating-keys can be discouraged while less powerful miners are still willing to participate.

Changing notes of txs

This seems inevitable. The only two ways I can think of is 1) excluding notes of txs, and 2) exclude notes from the input of txs' hashes. Not sure if there are more practical ways.

Changing different sets of txs

This also seems inevitable. Miners have freedom to choose their own sets of txs. Maybe miners earn less tx fee from doing this?

Conclusion

We construct non-parallelisable mining from Continuous Verifiable Delay Functions. It prevents miners from accelerating mining by having numerous mining hardware. It's easy to combine this with existing non-outsourceable mining and achieves very strong egalitarianism. There are still some known problems that give space to parallelise mining. We leave this as future work.