TLDR

: We show how to create timelock puzzles targetting RSA VDF ASICs with a hardcoded modulus.

Context

A key design decision for RSA VDF ASICs is whether or not the modulus should be hardcoded or programmable. There are reasons to have it hardcoded. These include reduced latency, power consumption, cooling, die area, complexity, and cost.

The main argument for a programmable modulus is that Rivest-Shamir-Wagner timelock puzzles require a programmable modulus (to be chosen by the puzzle creator). In this post we present a modified timelock scheme which works with a fixed modulus.

Construction

Let N

be a fixed modulus of unknown factorisation. Let t

be a time parameter. Assume that $2^{2^t} \mod N$

(i.e. the VDF output for the input 2

) is known as a public parameter (see next section for construction). A puzzle creator can now uniformly sample a large enough (e.g. 128-bit) secret s

and propose $2^s \mod N$

as a puzzle input.

Notice that the corresponding output $\big(2^s\big)^{2^t} = \big(2^{2^t}\big)^s$

can easily be computed knowing s

. Without knowledge of s

the input $2^s \mod N$

is indistinguishable from random, and therefore the corresponding output must be evaluated the “slow” way with repeated squarings.

Public parameters

It remains to show how to compute the public parameter $2^{2^t} \mod N$

. For small t

it suffices for anyone to run the computation and share the output. For large t

, if an MPC generated N

, the MPC can be extended to also generate $2^{2^{2^i}} \mod N$

for a few i

. For example, if the RSA VDF ASIC runs at 1ns per modular squaring then choosing $i = 52, ..., 60$

would allow for decade-long timelock puzzles.

Edit

: Notice that the construction also allows to do timelock puzzles with class groups.