Short RSA exclusion proofs for Plasma Prime

Some parameters of our plasma

The base data type for the amount is uint48

.

The segment size for each fungible asset is 2^40

.

First segment [0, 2^40-1]

is ether. The multiplier is 1e13

, so 1

in plasma is corresponding 10000gwei

in mainnet.

In addition, up to 256 types of assets (including ether with higher multiplier) may be included into plasma.

Prime set: each coin is corresponding to two prime numbers and each dust coin is corresponding two prime tree nodes.

The prime gap for 2^{50}

elements is lesser than 916. 60

bits are enough to store 916 * 2^{50}

. It is enough to use 64bit

prime numbers.

Abstract

The similar prooving schema was proposed by@vbuterin here. For arbitrary segments effectivity of this schema is O((\log N)^2)

. For \log N \sim 50

we need to send about 2500

64 bit numbers to make inclusion proof in plasma. This is not cheap operation and it will be rejected due to the high gas cost by the Ethereum mainnet. Also, operation with a prime number at the Ethereum mainnet are not cheap.

Here we propose a special game helping us to prove valid exit from plasma without presenting inclusion proof at the beginning of exit procedure but using short RSA exclusion proofs.

\DeclareMathOperator{\Hash}{Hash} \DeclareMathOperator{\Prime}{Prime} \DeclareMathOperator{\included}{included} \DeclareMathOperator{\ansestors}{ansestors} \DeclareMathOperator{\descendants}{descendants} \DeclareMathOperator{\inclusionprimes}{inclusionprimes} \DeclareMathOperator{\inclusionnums}{inclusionnums} \DeclareMathOperator{\exclusionprimes}{exclusionprimes} \DeclareMathOperator{\inclusionproof}{inclusionproof} \DeclareMathOperator{\exclusionproof}{exclusionproof} \DeclareMathOperator{\True}{True} \DeclareMathOperator{\False}{False}

Single element proving schema

Let’s define a \in [g, A]_{RSA}

if and only if \exists x: g^{ax}=A \mod N

, where N is RSA divider.

Simple consequence is that a \notin [g, A]_{RSA}

, if and only if \exists y: \gcd(y, a)=1

, a \in [g, Ag^y]_{RSA}

.

Let’s define s={a_i} \subset [g, A]_{RSA}

if and only if \forall a_i \in s: a_i \in [g, A]_{RSA}

.

To prove inclusion of single element a

we use proving schema proposed by Wesolovskihere:

Let’s define

$$h = g^a \mod N,\ B=\Hash(g, A, h),\ b = h^{x \div B} \mod N,\ r = x % B.$$

Than proving key is {b, r}

and it is enough to check following equation to prove the inclusion:

b^B h^r = A \mod N.

To prove the inclusion of multiple values, it is enough to prove the inclusion of multiplication of these values.

Nested short RSA exclusion schema

Let’s we have RSA exclusion proof

a = \prod a_i \notin [g, A]_{RSA}.

It corresponds

a \in [g, Ag^y]_{RSA},\ \gcd(a,y)=1.

If we have about 2500

multipliers of 64bit primes per output, the cofactor y

is about 160000

bit length and this is not useful for EVM.

As we can see below, in practice we need to prove the exclusion of one prime number. But if we store accepted from the operator proofs separately, we have huge overhead offchain, \sim2400

bits per prime vs 64

bits per prime if we store the proof batched.

We propose the following schema for short proof of exclusion of a single prime.

a \in [g, Ak]*{RSA}, \ a_i \notin [g, k]*{RSA}, \ a_i \in [g, h]_{RSA},

where

k = g^y \mod N,\ h = g^{\prod a_i} \mod N.

From the first equation we got following

b^B h^r = Ak \mod N,\

Indeed, to prove exclusion for a_i

, we need to prove that a_i \notin [g, k]_{RSA}

. That means that \gcd(y,a_i) = 1

. The second part is to prove that a_i \in [g, h]_{RSA}

. It is necessary in order to show that a_i

is part of a

.

So, we need to check two additional equations.

b_1^{B_1} g^{a_i r_1} = k g^{y_1} \mod N,\ 0<y_1<a_i,\ b_2^{B_2} g^{a_i r_2} = h \mod N.

Proving key contains {b,r,h,k,b_1,r_1,y_1, b_2, r_2}

11072

bits or 43.25

uint256

variables. It is not problem, because we are going to use it at the 2nd stage of exit game.

Aligned slice proving schema

Let’s define aligned slice as slice [x, y]

, where y-x = 2^t

and x = 2^t l

, where t

and l

are natural numbers. To describe the proving schema the binary tree of aligned slices is a useful representation.

For further use, we associate each tree node with two prime numbers.

.

This tree is corresponding to all aligned slices inside the segment [0, 8)

. For example, pair (5, 7)

is corresponding slice [0, 4)

.

Let’s define inclusion of aligned slice [2,4)

as an inclusion of following primes:

.

The left point means that the corresponding slice is included completely. The right point means that at least one left point inside the current node or the ancestors is included.

Then we can define exclusion of aligned slice [4, 6)

as the exclusion of following primes:

.

As we see, inclusion and exclusion proof affect one slice, the proofs must conflict at only one point. At the figure below we see the conflict between inclusion proof of [2,4]

and explosion proof of [4, 5)

at prime 23.

.

Let’s determine inclusion and exclusion proofs through auxilary functions \alpha

and \beta

that is applicable for each aligned slice P

of the tree and defined in the following way:

\alpha(P) = \included(P) \bigwedge_{Q \in \ansestors(P)}! \included(Q)

\beta(P) = \exists Q \in \descendants(P) \cup P: \alpha(Q)

\alpha(P)

corresponds to the inclusion of the left prime and \beta(P)

corresponds to the inclusion of the right prime.

Than define inclusion and exclusion proof for the aligned slice:

\inclusionproof(P) = \alpha(P) \bigwedge_{Q\in\ansestors(P) \cup P} \beta(Q) = \inclusionproof(\prod_{R \in \inclusionprimes(P)} R),

\exclusionproof(P) = !\beta(P) \bigwedge_{Q\in\ansestors(P)} !\alpha(Q) = \exclusionproof(\prod_{R \in \exclusionprimes(P)} R),

where \inclusionprimes

and \exclusionprimes

correspond the set of primes used for the proof ({3, 7, 23, 29} {2, 11, 37}

for examples above)

As we can see, the operator cannot present inclusion proof for any included small coin. We define smaller intervals included if the operator present inclusion proof for any interval containing the one.

It is obviously to check, that

\exclusionproof(P) \Rightarrow !\inclusionproof(Q), \forall Q \in \descendants(P) \cup P.

\inclusionproof(P) \Rightarrow !\exclusionproof(Q), \forall Q \in \descendants(P) \cup P.

The main difference between this construction and proposed by @vbuterin here is that we are using only inclusion proof to prove inclusion of aligned slice and only exclusion proof to prove exclusion of the aligned slice. This approach helps us to batch inclusion and exclusion proves for arbitrary segments, as you can see below.

The exit game

The first enumerate all prime numbers used in the proof and determine

\Prime(N) = \max {p \in \Prime: p\leq 1024 N}.

\inclusionnums

is the set of natural numbers corresponding \inclusionprimes

.

Here is the game determining the validity of exit E

. We use some components from PrimeHash game.

The fist the exiter publish segment S

. This segment determine aligned slices and included primes. If the challenger has exclusion proof coresponding S

, he can compute conflict primes and prove exclusion of one. It is enough to reject the exit procedure.

[.

See the figure with better resolution.

](https://raw.githubusercontent.com/snjax/drawio/master/rsagame2.svg?sanitize=true)## Bibliography

[Plasma call #17

](https://www.youtube.com/watch?v=YjTF05SeYxo)

@vbuterin, RSA Accumulators for Plasma Cash history reduction

@vbuterin, Log(coins)-sized proofs of inclusion and exclusion for RSA accumulators

Benjamin Wesolowski, Efficient verifiable delay functions

@snjax, PrimeHash game for Plasma Prime