By @keyvank and @irnb

While researching on privacy solutions and applications of ZKP, we discovered a technique, by which people can burn their digital asset (E.g ETH) by sending it to an unspendable address, and later build a ZK proof showing that some amount of tokens has been burnt in a transaction in an older block, without revealing the transaction. Why is it important? People can do plain EOA-to-EOA transfers which privately have impact in some other verifier smart-contract on the system.

Recap on Elliptic-Curve digital-signatures:

In Elliptic-Curve based digital signatures, normally there is a secret scalar s

, from which a public-key is deriven (By multiplying the generator point with the scalar: s \times G

)

A public-key is spendable if and only if its corresponding private-key s

exists. We can generate unspendable public-keys by generating random points on the curve. But how can other people be sure that a point is indeed random and not the result of calculating s \times G

? We can generate points that are very unlikely to be spendable by using fancy-looking patterns in their x

coordinate. E.g: In case of Secp256k1 we can pick x=123456789

and calculate y

by putting it in the curve equation:

y^2=x^3+7

Because of the discrete logarithm problem, it's practically impossible to calculate s

where s \times G

is equal with our point.

Let's say R

is an unspendable public-key (Let's call it the reference unspendable public-key). People can publicly burn their tokens by sending them to R

, after doing so, others can indeed conclude that they have burnt their tokens, because they can all see the transactions and they know that R

is unspendable.

We can derive infinitely many provably-unspendable public-keys from a reference unspendable public-key

Let's pick a random secret value t

and calculate a new point D=R + t \times G

. We can prove that D

is also unspendable, because log_G(D)=log_G(R + t \times G)=log_G(R) + t

, and since we can't calculate log_G(R)

, the public-key D

is also unspendable.

Obviously, D

is a completely new point that does not seem unspendable. We can convince others that D

is unspendable by revealing t

, because then people can verify that D

is the result of adding some other point to R

.

Using the help of Zero-Knowledge proofs, we can hide the value of t

We just need to prove that we know a secret value t

where R + t \times G == D

. We can go even further. Given that we have access to the previous block hashes in our Ethereum smart-contracts, we can prove that some EOA-to-EOA transaction has happened in the previous blocks, burning some amount of token (It actually doesn't need to be an EOA-to-EOA, and it also could be a ERC20-transfer).

Are there any applications?

We honestly are not sure whether there is an application for such a proof, but here are some random thoughts:

Imagine there is a ZKP verifier contract that verifies if someone has burnt some amount of USDT in the previous blocks (Nobody can't detect that since it's a very normal looking ERC-20 USDT transfer). Imagine we mint an equal amount of BUSDT (Burnt/Backed USDT?!) in case someone proves that he has burnt USDT (We prevent minting the same burnt coins again using the help of nullifiers). The number of BUSDTs in circulation will be equal with burnt USDTs and no one can find out the burners of those USDTs (Unless their reveal their secret t

). So we can claim that BUSDT is secretly backed with USDTs, and thus backed with actual USD, with the difference that it can't be frozen by the company behind the stablecoin.