Proof of Burn Challenge Notes

Authors Note:

This is great fun, it's like learning about computers all over again! I'm young enough to remember trying to run C without compiling it and failing to grasp why it wasn't working. Granted, I was about twelve years old at the time (Visual Basic was my specialty at that point). I'm actually finding it extremely insightful learning about the various constructions of the payment addresses - there is a good reason for this[1].

I spend most days continuing to learn Haskell, writing docs, re-learning math, revisiting cryptography and you know what, I have rediscovered my passion! I lost the passion when I started working as a web developer and was pigeon holed into: we must do X, we must do Y, I have likely learnt more during the Plutus Pioneer Cohort than I did during the last 12 months of my formal employment.

UPDATE: So, we can scrap this idea, the documentation I was reading failed to mention that ADDR was not a hashed public key, it's actually quite a complicated Byron legacy address. In fact, from what I understand it was a little over-engineered, thus has become legacy with the introduction of Bech32. However, that doesn't mean we can't dive a little deeper into whether or not it is possible to generate a burn address using the original scheme legacy scheme, but by modifying part of the payload. I've not yet read too much about the consensus algorithm and whether or not it still supports super legacy 'wallet addresses'... Regardless, PoB is suppose to be implemented as a smart contract, but I'm gaining a wider understanding of the system by reading all this documentation. So I think I'm going to keep doing this for now...

Synopsis

~~Currently working on the PoB Challenge, for which I actually have some questions... Concerning backwards compatibility: can we use P2PKH style addresses in Cardano? Since the paper uses their Random Oracle Model to demonstrate indistinguishability between burn addresses and P2PKH addresses... At least, that is what I took from it. I've been stumbling across all kinds of old Cardano docs (Cardano SL) and I'm curious, if we generated our own address using:~~

1:

$$ADDR_{init} = base58(ADDR_{hf} \;||\; CRC32(ADDR_{hf})) \;;\; \exists\, ADDR_{hf} = hf(k_{pub}) \mid k_{pub} \in K_{derivation\ scheme}$$

~~Then, as 'hf' is a hashing function that provides a public key hash, we can modify the initial public key address, to create our burn address:~~

2: $ADDR_{burn} = ADDR_{init} \mid ADDR_{hf} = \left[hf(k_{pub}) \oplus 1 \right]$

~~This means our 'silly person who wishes to burn ADA' can create a UTxO from the init address, which only he/she has the private key to (only he/she can spend that UTxO, as wallet mnemonics are unique, meaning only his/her seed could create that public key address and only he could have the secret key, unless he/she shared it), then he (or she, sorry!) could create a transaction on the blockchain by signing over the entire value of the UTxO to the burn address.~~

Right...

~~If my thinking is correct, this should still produce an indistinguishable key if one was to use the Random Oracle Model outlined in the PoB (adapt P2PKH to... I believe it's Byron encoded addresses? Base58?) paper. Furthermore, it would make the final address non-spendable, since there would be no matching private key to sign the UTxO. Thus, ADA = burnt!~~

~~You may be wondering: why in the hell would you go to all this hassle when IOHK have implemented the same mechanism through the use of a single tag? Again, my thinking is: if one is, as Charles put it, stupid enough to request the burning of ADA, they should burn their own ADA. So, in addition to being able to verify the address is a burn address, personally, I would also like to see one more property added to the list of requirements for Proof-of-Burn, this would be: Individual Verifiability:~~

Research

Links:

- https://github.com/input-output-hk/cardano-docs/blob/master/docs/cardano/addresses.md
- https://github.com/cardano-foundation/CIPs/blob/master/CIP-0019/CIP-0019.md
-

Footnotes

1. I did pretty well in compilers and ARM assembly programming modules during my undergraduate degree. These various address schemes (P2PKH, P2SH, Byron Legacy Base58, etc) remind me of writing ARM assembly, polling for IO by checking individual bits instantly springs to mind, it was great fun! In addition, I do think there is a distinct advantage to gaining a broad understanding of an eco-system (or at least trying to, you'll get things wrong to begin with, but you'll learn with time, I'm talking very much from my own experience).

2. I've seen OLD documentation to say that Cardano addresses have been built like this in the

past, however, I have no idea if this address formatting is still compatible with the current version of Cardano, and if ouroboros would verify this transaction?