Prototyping the Blockchain Lawnmower (Code Companion #3)

Author	josh
Date	2020-02-17 13:09:57

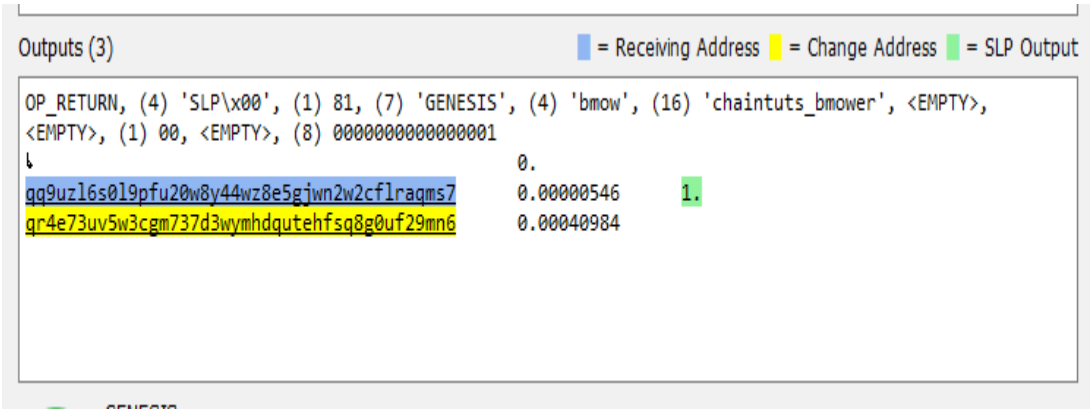
Overview

In my recent tutorial on tokenization, I showed a possible future where we might trade assets like homes, cars or even lawnmowers on the blockchain. I built a fun prototype to demonstrate this concept - using a Bitcoin Cash token, a Raspberry Pi, and my trusty lawn tractor! This tutorial is a technical look at how I developed that prototype, and how more advanced versions might be built.

Building the Blockchain Mower

Bitcoin Cash SLP Tokens

The first step in this process was to create a non-fungible token on a real blockchain that can represent my tradeable lawn tractor. For this, I felt the Bitcoin Cash blockchain would be the way to go. BCH offers a token standard called SLP (Simple Ledger Protocol) for both fungible and non fungible tokens. Using the Electron Cash SLP wallet, I issued a new token called `chaintuts_bmower`. This was simple through the wallet's interface - I simply needed a little bit of BCH in the wallet for the transaction fees. SLP tokens are created using `OP_RETURN` transactions, which allow placing arbitrary data on the chain.



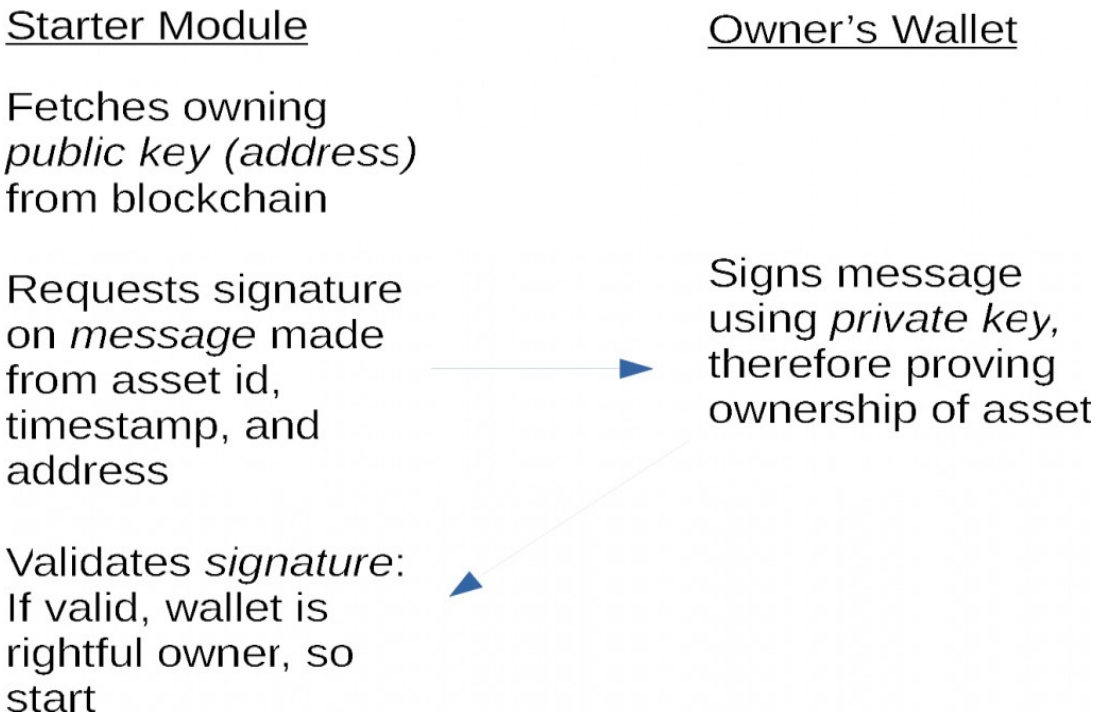
Issuing an SLP token with an OP_RETURN transaction

Now instead of using the full non-fungible token standard, I simply issued a token with a cap of 1 that is non divisible. So there's only ever one mower token representing my real-world mower that can be traded around on SLP addresses. If you had a full fleet of mowers to track, you'd have to use a more advanced issuing standard.

Digital Signatures with Electron Cash

Now that I had a token at an address I owned, I needed to create the code that would prove and validate ownership of the asset. We have a token at a BCH SLP address, meaning that the asset is owned at a *public key*. The address owner has a *private key* for that address that proves ownership of the asset. So, the way that we can have a system where a starting mechanism validates the rightful owner of that asset, we can use a challenge-response mechanism based on *digital signatures.*

The asset is owned by a *public key* on the blockchain. That address has a *private key*. The starter knows the public key, but can't know the owner's private key for obvious security reasons. So, the starter module will request a *digital signature* from the owner that proves they have the private key. If the signature is valid, that means they are indeed the rightful owner of the asset!



Now how did I implement the code to do this? I went with the trusty code of Electron Cash SLP to help me generate and validate signatures. I trimmed down the library code as much as possible to allow me to import the modules I need directly in Python, rather than calling Electron Cash's command line calls directly. This makes the code much faster and cleaner.

The client code for this prototype is a simple Python script that runs on a USB drive. This excerpt shows the signature generation:

```
# Fulfill the signing request using the owning key
def sign_request_electron(owning_address, message):

    signature = ""

    # First, fetch and decode the WIF key
    privkey_wif = load_privkey_file()
```

```
    _, privkey, compressed = deserialize_privkey(privkey_wif)
    ec_key = regenerate_key(privkey)

    # Create an EC Key instance and sign the message
    signature = ec_key.sign_message(message, compressed)
    signature = base64.b64encode(signature)

    return signature
```

On the starting module end, the signature is requested and verified:

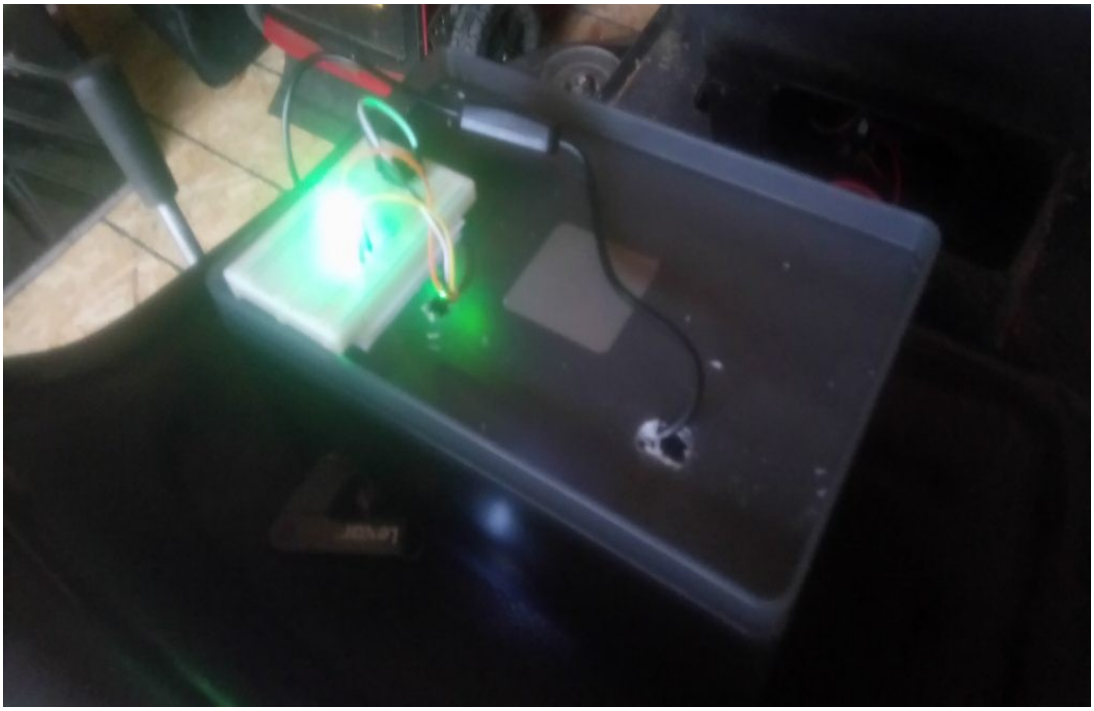
```
# The driver function that generates a signing request, sends it, and verifies
the result
def generate_and_verify():

    asset_id = owner.load_assetid()
    owning_address = owner.load_owner()
    timestamp = time.time()

    req_string = generate_message_request(asset_id, owning_address, timestamp)
    print(f"Generated signing request for\nAsset ID: {asset_id}\nOwned by:
{owning_address}\nTimestamp: {timestamp}\nSigning request message:
{req_string}")

    signature = send_signing_request(owning_address, req_string)
    print(f"Signature returned by device: {signature}")

    verified = verify_message(owning_address, base64.b64decode(signature),
req_string.encode("utf-8"))
    if verified:
        print(f"Signature successfully verified! Asset is owned by
{owning_address}")
    else:
        print("Signature is invalid. Cannot prove asset ownership")

    return verified
```

Actually Starting The Mower

This was the trickiest part...figuring out how to electronically control my lawn tractor and bypass the ignition switch needed much more thought and work. At first I thought that I might try to touch two bypass wires together using a solenoid, but this seemed a bit of an "unclean" solution. Then, I learned about the concept of relays!

A relay is an electronically controlled switch. On one end, small wires connect to the Raspberry Pi. On the other circuit, you can open and close a circuit. I connected that circuit from the lawn tractor's battery to the starter solenoid signal terminal.

So, when the signature is verified by the owner from their USB "wallet", I had the code light up a green LED indicating the push-start mechanism is live. Then, the user can press a button to activate the relay, closing the starter circuit and turning over the mower!



Breadboard with indicator green LED and push button. Private keys, push to start!

The Blockchain Mower Prototype - Lots of Parts and Fun!

This project was a blast to build! There's lots of details, but the general idea is fairly straightforward. A token represents a real world asset on the blockchain. This asset has an address (public key). The user's wallet can prove ownership of the private key for that asset with a *digital signature.* If the signature is valid, the "starter module" knows you're the rightful owner, and will start up for you!

In the future, this can apply to more interesting projects than just a lawn tractor. This is a fun prototype, but the real use cases are with assets we currently title and trade through government agencies. Imagine trading your home on the blockchain, with your front door unlocking through cryptography. Or imagine selling your car as a token, and tapping your phone wallet to start it! The possibilities are exciting.