Intro

How do we give cryptocurrancy to people who don't have computers (or smart phones)

We need a way to sign transactions. The cheapest way is NFC cards which cost between ~1.5 USD. With discounts for bulk purchases. NFC cards is a kind of trusted hardware. That are used everyday as debit cards and metro cards.

The fact that this hardware is ubiquitous allows us to benefit from economies of scale and produce these cards at very low costs.

Previous work

Using trusted hardware for physical payments has been done before.

1. Opendime

Locks the private key inside a hardware wallet. The funds can only be spent if you break an anti-tamper strip. These hardware wallets are not reusable once they have been withdrawn. Its also prohibitively expensive to use them for lower price transactions as the hardware currently costs ~ 45 USD

1. Kong

Is a independent currency that is locked inside the kong hardware wallets for a a period on the order of years.

Here we propose hardware-notes. We harden NFC cards with a smart contract so that it can become a reliable medium of exchange in areas with intermittent network connection. Where the physical hardware is passed. And possession of the hardware-note is required to withdraw the cryptocurrancy that is deposited to that card.

We decentralize the role of hardware manufacturer and allow users to accept any hardware that they deem trustworthy. We also allow erc20 support so anyone can create any currency inside these cards.

Smart contract

Manufacturer Card Registry

We create a smart contract that a manufacturer lists the cards they have created.

The public key of each card is stored in a merkel tree where a leaf is the public key. The private key is locked inside the card.

Deposit

A smart contract allows users to deposit cryptocurrancy to any card.

They send a transaction with

deposit (address erc20, uint amount, uint withdraw_delay ,address card_pub_key)

A single card can have multiple currencies with various denominations.

Withdraw

There is a two step withdrawal process.

Both steps the withdrawer proves possession of the hardware-note by signing one of the last 256 block hashes.

Signal Withdraw

First the user signals they want to withdraw.

signal_withdraw (bytes32 blockhash, address pub_key)

The first step is followed by a wait period. The wait period is variable. It is defined when a deposit as withdraw_delay

was made to that hardware-note.

Execute Withdraw

After the wait period a user can withdraw the funds by again signing one of the last 256 block hashes.

The wait period is included here to allow people to accept payments if they are operating with an outdated state. As long as their state.age < coin.withdraw_delay

they can be sure that they will at least be able to call execute_withdraw

for the coin they receive.

Spending Hardware-notes

When deciding on a hardware-note value the app checks

1. That the card has been issued by made by a reputable manufacturer.
2. The currency and amount that the card holds.
3. That the state the app is working with is less than the withdraw delay for that coin.

If all of these are true the merchant can accept the coin for its defined value.

State Updates

Every once in a while a merchant will need to update their state. This means going online and downloading a list of coins that have been

1. Withdrawn
2. Are in the process of being withdrawn
3. Have been deposited
4. They also need to download the list of new cards from their trusted manufacturers (this can happen much more rarely)

The frequency of state updates required is equal to the min withdraw_delay

that they want to accept. So this can be tuned by the merchant who can refuse to accept a card with a short withdraw_delay

Example usecase

Simple Payment

1. I decide to move to country X where the annual inflation rate is 10,000 %
2. I buy a bunch of empty hardware-notes from the most popular manufacturer that merchants accept in country X.
3. I load one hardware-notes with 5 USD the other with 15 USD worth of a stable coin. Both with a 14 day withdraw_delay
4. I arrive and go to a shop in the country.
5. I buy 10 USD worth of food.
6. I pay with the 15 USD hardware-note to the merchant.
7. The merchant checks the hardware-note with the app and agrees to accept it with 15 USD value.
8. The merchant returns to me a 5 USD hardware-note or 5 USD in local currency.

Here we assume the merchant has updated their state since I deposited into my crypto-currancy into the hardware-note.

Merchant

1. I am a fruit seller in country X.
2. I sell fruit every day and had to accept payment in currency that experiences 10000% inflation
3. I cannot accept hardware-notes in my business because my average transaction size is less than 1 USD where it is uneconomical to create hardware-notes for such a small denominations.
4. So I operate in the local currency as soon as I have earned 10 USD I exchange it to a hardware-note.
5. I use hardware-notes to buy my supplies.

Attacks

1. NFT war driving

If i get close to someone while they hold these hardware-notes i can steal from them. I can ask the hardware-notes to sign the latest hash. Broadcast it to perform signal_withdraw

Then withdraw_delay

seconds later I can find that same person again and calling execute_withdraw

.

This attack is mitigated by

- Having more than one hardware-notes in my pocket. Its really hard to read a single one
- Holding my hardware-notes in a Faraday cage, tin foil package
- Lying merchant

When I try and spend my hardware-notes a merchant is always able to lie to me and say that the balance is not what I claimed it to be.

I can overcome this by going to another merchant and checking again with them.

In smaller communities this attack is not possible as lieing merchants will soon be found out. In bigger communities people will hopefully have a choice of who to transact with.

But we assume there is a single honest check balance person available.

1. Withdraw instead of check balance

Instead of checking a users balance a merchant can begin the withdraw period for that user. They would then need to reject the coin and re-scan it again after delay

seconds in order to complete the withdraw.

To overcome this we can add a long time delay for the withdraw

function on the hardware-notes.

For example a simple check balance can take 1 second. In order to perform a withdraw we could add a delay this to 10 seconds.

If we build a strong expectation that a user only needs to scan their coin for 1 second to check its balance then we can avoid these kinds of attacks.

1. Rouge manufacturer

A manufacturer claims to have locked a given public key in a hardware-note but in fact they have not and this public key is in the clear.

The manufacturer will then be able to double spend the funds held on this hardware-notes.

You have to trust the hardware manufacturer at the time of hardware-notes manufacture.

We hope that an ecosystem of hardware manufacturers exist to allow this so that users have an option on which manufacturers to trust.

Its important to note that this is the same with all hardware that holds crypto-currancyies. You have to trust the hardware you are using. And there is no way to economically validate that your hardware is correct.

Conclusions

We allow creation of physical crypto-currency assets with variable timeouts. This can allow use of crypto currency in areas that do not have consistent internet connection.

This reduces the time that you need to be online which will allow communities without reliable internet / electricity to use crypto currencies.

Creation of coins is completely open so any actor can create currencies and use this infrastructure to transact with them.

These currencies can be converted back to digital form after waiting a delay period which means you can move between the physical and cryptocurrency world.