ツールを作りかけたのでわかる bitcoin HD Wallet のしくみ

sanemat {AT} tachikoma.io

- Hierarchical Deterministic Wallet 階層的決定性ウォレット
- BIP0032/BIP0044

これはなに

- Wallet の UXTO (未使用残高) を wallet 実装無しで計算するツールを作ろうとした
- 現在進行形で作りかけ
- 結論は「既存の Wallet を使え」なんだけど、 そのツールを作る過程で bitcoin HD Wallet のしくみを理解してきたので、まとめました

あやふやなところもあるからわかりにくかったら突っ込んでね

アドレスの固定

図 1: shiba-shop

さっき bitcoin 専門社内おやつ商店の話で、1 アドレスを紙に書いてしまった話をした。

アドレスは使い捨てにする

- ビットコインアドレスって、使い捨てにしたほうがいい。
- こちらの取引金額なんてバレてもいーじゃんってはじめは思う。
- こちらのセキュリティ的な話。
- 相手のセキュリティ的な話。
- 51%攻撃受けやすくなる。
  - transaction を pool に入れたり、block の中だったりでゴニョゴニョしうる余地を産んでしまう
- 表示側もワンタイム生成のアプリにすべきですな。

Wallet の UXTO (未使用残高) を wallet 実装無しで計算するツール

- なぜ wallet 実装無しでやりたかった?
- 着金したかどうかってどうやってわかるんじゃろう
- 残高どうやって出そう
- wallet 使えばいいじゃん
- HD Wallet にして、拡張公開鍵だけ入れておけばいいのでは?
  - 一見良さそうに見えた。

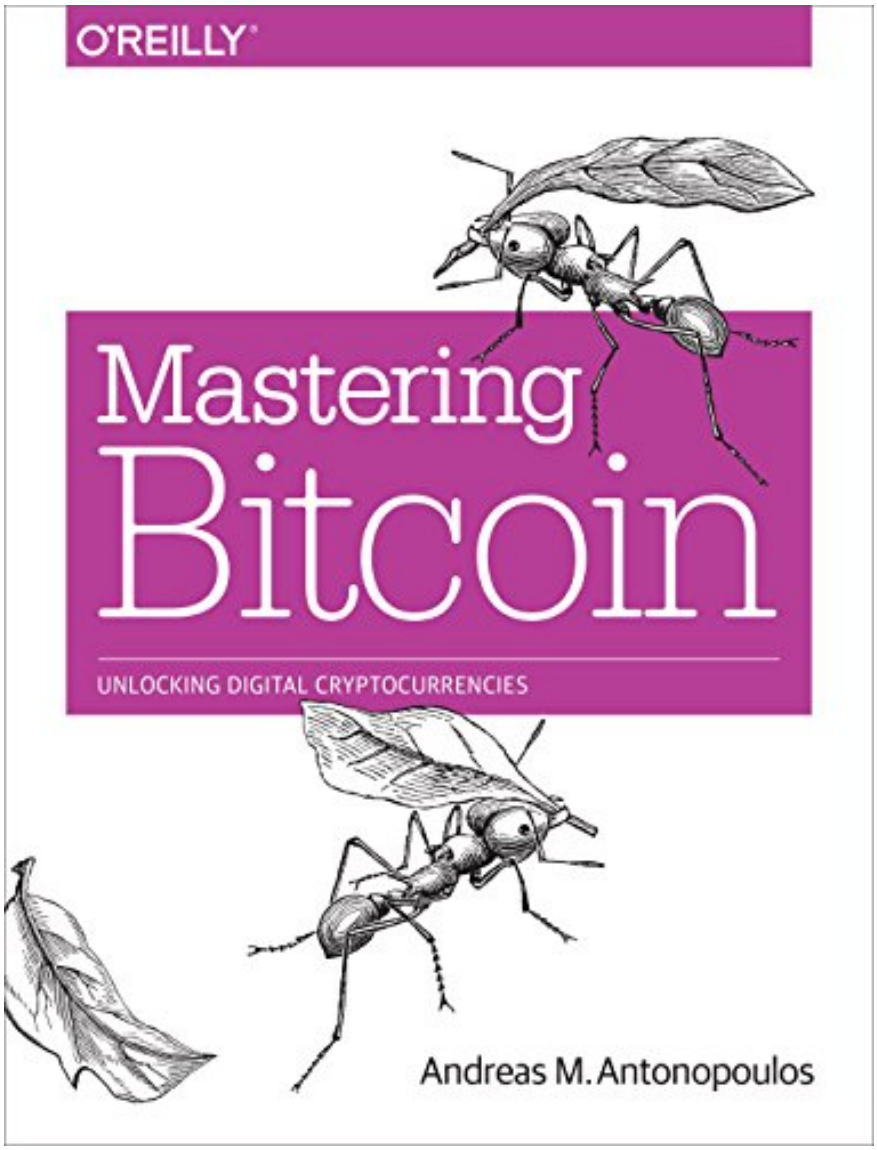
送金依頼のような

- アプリ側の実装によるのかもしれないが、copay android アプリからだと、拡張公開鍵だけで送金「依頼」のようなものが出せた。
- wallet client 側の見た目の数字は減って、秘密鍵を持っているアプリに、「この送金を verify しますか y/n」のようなダイアログが出て、y を押したら送金処理になった。
- もちろん n を押したり、無視したりすれば送金されないが、ちと、ちと。

HD Wallet と BIP0032/BIP0044 の説明

HD wallet(Hierarchical Deterministic Wallet) 階層的決定性ウォレット

mastering bitcoin を読むか、BIP0032/BIP0044 直接読んでくれ! 以上



ウォレット

- 公開鍵の別表現がビットコインアドレス

- 秘密鍵をどう持つか/作るか、のはなし

ランダムウォレット (非決定性ウォレット)

- ウォレットはランダム生成した秘密鍵の束

決定性ウォレット

ウォレットで共通の seed から一方向ハッシュで秘密鍵生成

HD wallet(Hierarchical Deterministic Wallet)

13 個の mnemonic code から seed を作成 seed から master key 生成、そこからツリー上に child keys, grandchild keys と生成

(mastering bitcoin からコピペ)

1. ツリー構造に情報付加できる

key の path こんなやつ `Bitcoin second change second m / 44' / 0' / 1' / 1 / 1`

2. ユーザーが秘密鍵に触れることなく公開鍵を生成できる

ビットコインアドレスを作れる、と同じ意味合い

気になるところ

拡張公開鍵が chain code を含んでいる

拡張公開鍵って「公開鍵」って言いつついろいろ出来てしまうので、ふつうに ssh の公開鍵みたいに全世界に公開していいの? って問題意識とも関連してくるんだけど

> 拡張公開鍵は chain code を含んでいるため、もし子秘密鍵が知られているまたは漏洩してしまった場合、この chain code を使ってその他すべての子秘密鍵を導けてしまうこのリスクへの解決策として、HD ウォレットは hardened child key derivation 強化子公開鍵導出 関数を使っているもし拡張公開鍵の利便性を使い、しか

> も chain code の漏洩リスクを回避したいのであれば、通常の親 (親公開鍵) ではなく、強化された親 (親秘密鍵) から拡張公開鍵を導出すべきです。ベストプラクティスとしては、マスターキーの 1 階層目の子供を常に hardened derivation を通して導出されるようにしておくことが良いでしょう。

なんか小さくベストプラクティスが書いてあるが、一般に使われるクライアントは、みんなこれやってると思っていいんだろうか…

UTXO

UTXO (unspent transaction output)

だいたいの wallet で wallet.balance() とかでとれるやつ

ここまで来てやっと意味がわかった。

- wallet は未使用含めて bitcoin アドレスを持っていて (作成できて)
- それには path に 0 とか 1 とか 2 とか順に名前がついてて
- path は小さい順から使う暗黙ルールがあるらしい (未確認)
- なぜなら、未使用が 20 個続いたら探索打ち切り、などがウォレットアプリ側で実装されている
- なので、条件に依っては、あるウォレットで見つからなくなってしまった bitcoin が、アドレスの使用数が進んだら出てくることもあるようだ (未確認)

まだわからないところ

- ツリー状のはどうたどるんだ?
  - path のルールがコレ BIP: 44, Title: Multi-Account Hierarchy for Deterministic Wallets
- これが bitcoin-explorer って名付けられてるタイプのライブラリの仕事なのかな (未調査)

まとめ

俺たちの戦いはこれからだ!!

ツールを作ると学習が進む (作れてない)

参照

- Bitcoin ウォレットを実装する- ビットコインの仕組み：Bitcoin を技術的に徹底解説！
- Translations of Mastering Bitcoin pdf
- BIP: 32, Title: Hierarchical Deterministic Wallets
- BIP: 44, Title: Multi-Account Hierarchy for Deterministic Wallets

ヨタ話

プログラム言語は何で実装見てたんですか

Nodejs です

各言語の実装は bip からリンクが貼ってある

- golang 学習がてらやろうとしたら全然そんな余裕がなく、土地勘のある ruby と nodejs だと nodejs にライブラリが結構あった。
- bitcoin core が C++だし、C++が一番楽なのでは。俺書けないけど。
- それか python。俺書けないけど。
- btcd が golang だから golang もよさそう。俺書けないけど。
- ほんとは golang で bitcoin プロトコルしゃべろうと思ったんだ…

nodejs でなんとなく見たのは

- bitcoinjs-lib
- bitcore-lib (bitpay.io)
- bcoin (purse.io)
- この順番がリリース順で、API の洗練され具合も下が良い。
- あと、full node か SPV node か、web service との連携具合とかが違う。
- なんだけど、npm ソムリエ的には、おすすめは上から下。
- ただいずれにしてもそこまでアクティブではないように見える。
- 一応メンテはされてる。