grincon0
18.11.09 // c-base berlin

Grin Wallets

@yeastplume

Overview

1. Basic MW Transaction Workflow
2. Grin Transaction Workflow
3. MW Wallet Challenges
4. Current Grin Wallet Toolset
5. Community Efforts
6. Remaining Time Q&A

Basic MW Transaction

1. Pile of Outputs (In) equalling a certain amount are dumped into the partial TX
2. A pile of outputs are created that spend the entire amount
3. In - Out = 0 + Excess

TX passed around and signed off by all involved parties

All happens outside the chain

Simplest MW Transaction Workflow

Sender -> Recipient -> Chain

$$Y - Xi = (113*G + 3*H) - (28*G + 3*H) = 85*G + 0*H$$

1. Carol adds Input + Outputs + Rproof + Private key (Blind excess) + Amount to TX
2. Partial TX sent to Alice
3. Alice chooses Blinding Factor, creates her output, signs TX
    a. Needs value 28 in order to create excess signature
4. Node

Basic Challenges

- All transactions **must** be created interactively
    - Need to pass around partial transactions somehow
    - Private data needs to be hidden from other transactors and observers
    - ‘Potential’ Outputs within wallets need to be managed

- Positive Aspects
    - TX creation can happen offline, anywhere, in private
    - Can’t spam recipients unknowingly
    - Many creative possibilities

INTERACTIVE DOES NOT MEAN 'ONLINE'

(Can all be made to work)

Basic MW Transaction Creation

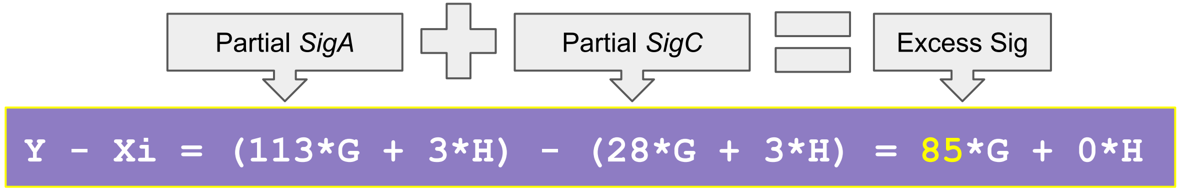
Basic Key sharing problem:

$$Y - Xi = \overbrace{(113*G + 3*H)}^{\text{Alice's Output}} - \overbrace{(\underbrace{28}_{\text{Carol's Private Key}}*G + 3*H)}^{\text{Carol's "Input"}} = \underbrace{85}_{\text{Excess Sig Value}}*G + 0*H$$

To construct + sign excess sig, recipient Alice needs values 113 and 28

Grin’s Aggsig TX Workflow

Grin’s alternate Aggsig workflow:



Y - Xi = (113*G + 3*H) - (28*G + 3*H) = 85*G + 0*H

- Participants sign for their parts of the excess
- Part sigs are added together at the end
- No private information is exchanged

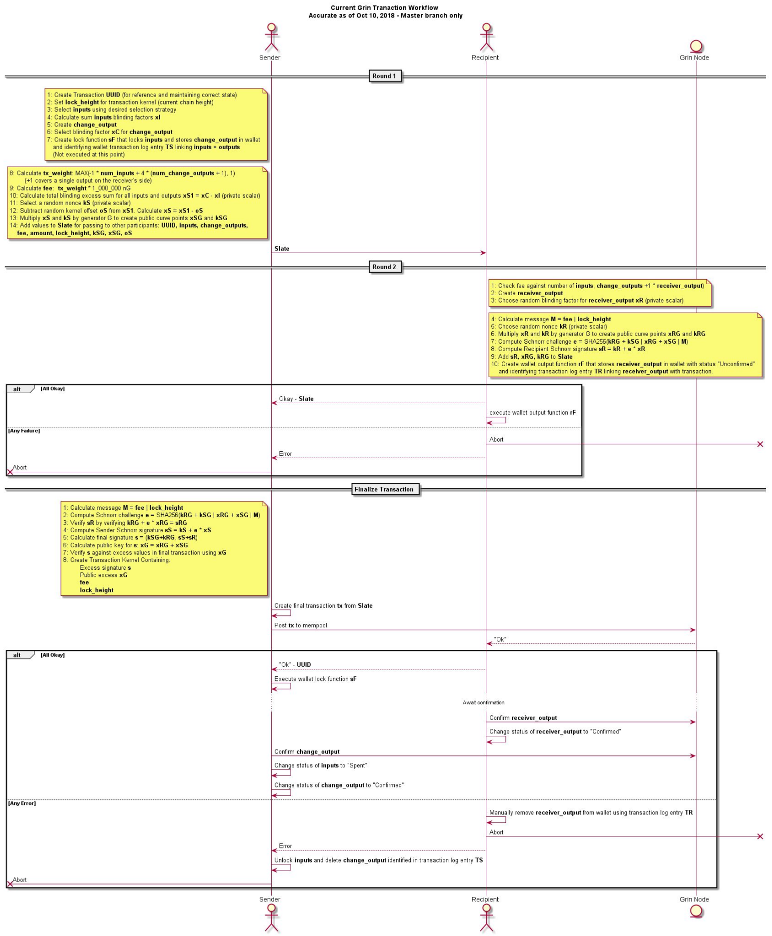
Grin Aggsig TX Workflow

- 2 Rounds Required
- 1st Round
  - All participants add outputs to ‘Slate’
  - All participants add PUBLIC (k*G) excess + schnorr nonce values to Slate
- 2nd Round
  - All participants add their excess signatures
- Anyone can then post TX
- Slate can be public (email enabled)
- Shortest route is (Sender -> Recipient -> Sender -> Post)

Challenges (Recap)

- Works great, but now it’s even more complicated
  - Fairly complex interactive exchange to manage, potentially among many parties
  - Lots of output / input management in everyone’s wallets

https://raw.githubusercontent.com/mimblewimble/grin/master/doc/wallet/transaction/basic-transaction-wf.png


Sender
Recipient
Grin Node
Round 1
Slate
Round 2
alt
[All Okay]
Okay - **Slate**
execute wallet output function **rF**
[Any Failure]
Abort
Error
Abort
Finalize Transaction
Create final transaction **tx** from **Slate**
Post **tx** to mempool
"Ok"
alt
[All Okay]
"Ok" - **UUID**
Execute wallet lock function **sF**
Await confirmation
Confirm **receiver_output**
Change status of **receiver_output** to "Confirmed"
Confirm **change_output**
Change status of **inputs** to "Spent"
Change status of **change_output** to "Confirmed"
[Any Error]
Manually remove **receiver_output** from wallet using transaction log entry **TR**
Abort
Error
Unlock **inputs** and delete **change_output** identified in transaction log entry **TS**
Abort
Sender
Recipient
Grin Node

Current Grin Tranaction Workflow
Accurate as of Oct 10, 2018 - Master branch only

1: Create Transaction **UUID** (for reference and maintaining correct state)
2: Set **lock_height** for transaction kernel (current chain height)
3: Select **inputs** using desired selection strategy
4: Calculate sum **inputs** blinding factors **xI**
5: Create **change_output**
6: Select blinding factor **xC** for **change_output**
7: Create lock function **sF** that locks **inputs** and stores **change_output** in wallet
and identifying wallet transaction log entry **TS** linking **inputs + outputs**
(Not executed at this point)

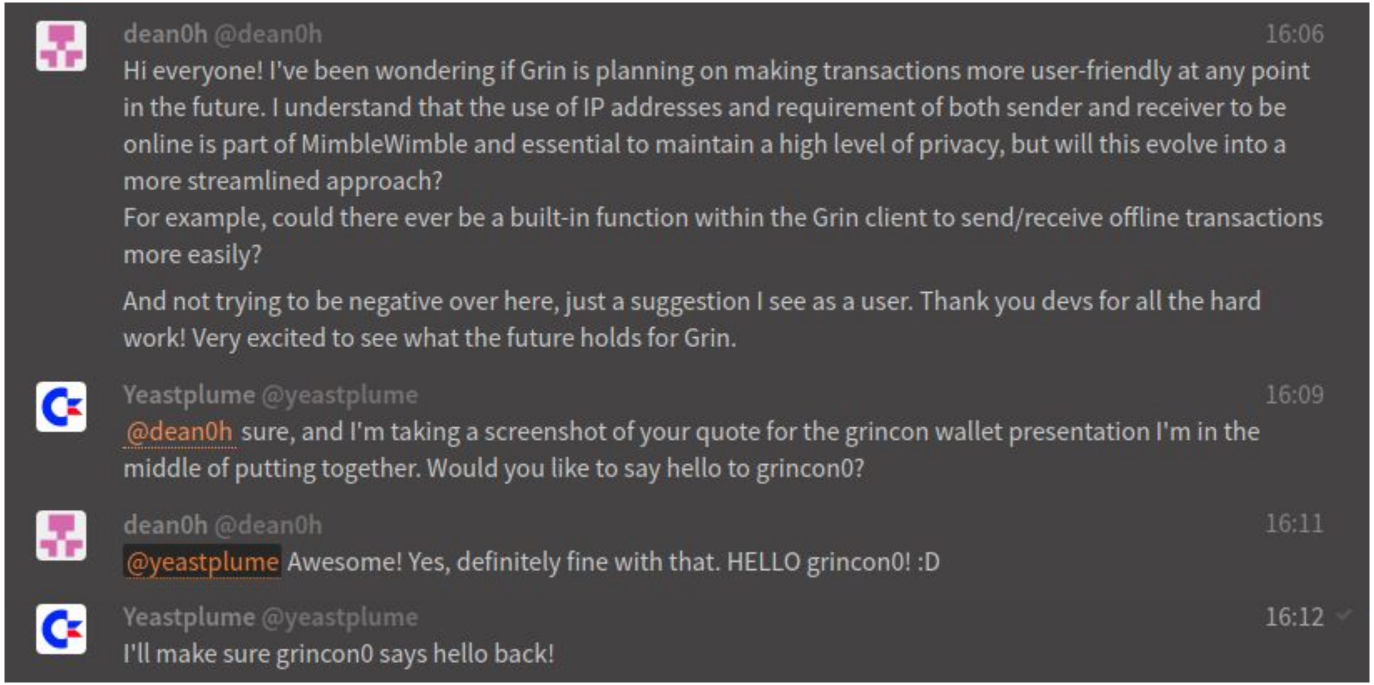
8: Calculate **tx_weight**: MAX(-1 * **num_inputs** + 4 * (**num_change_outputs** + 1), 1)
(+1 covers a single output on the receiver's side)
9: Calculate **fee**: **tx_weight** * 1_000_000 nG
10: Calculate total blinding excess sum for all inputs and outputs **xS1** = **xC** - **xI** (private scalar)
11: Select a random nonce **kS** (private scalar)
12: Subtract random kernel offset **oS** from **xS1**. Calculate **xS** = **xS1** - **oS**
13: Multiply **xS** and **kS** by generator G to create public curve points **xSG** and **kSG**
14: Add values to **Slate** for passing to other participants: **UUID, inputs, change_outputs, fee, amount, lock_height, kSG, xSG, oS**

1: Check fee against number of **inputs**, **change_outputs** +1 * **receiver_output**)
2: Create **receiver_output**
3: Choose random blinding factor for **receiver_output xR** (private scalar)

4: Calculate message **M** = **fee** | **lock_height**
5: Choose random nonce **kR** (private scalar)
6: Multiply **xR** and **kR** by generator G to create public curve points **xRG** and **kRG**
7: Compute Schnorr challenge **e** = SHA256(**kRG** + **kSG** | **xRG** + **xSG** | **M**)
8: Compute Recipient Schnorr signature **sR** = **kR** + **e** * **xR**
9: Add **sR, xRG, kRG** to **Slate**
10: Create wallet output function **rF** that stores **receiver_output** in wallet with status "Unconfirmed"
and identifying transaction log entry **TR** linking **receiver_output** with transaction.

1: Calculate message **M** = **fee** | **lock_height**
2: Compute Schnorr challenge **e** = SHA256(**kRG** + **kSG** | **xRG** + **xSG** | **M**)
3: Verify **sR** by verifying **kRG** + **e** * **xRG** = **sRG**
4: Compute Sender Schnorr signature **sS** = **kS** + **e** * **xS**
5: Calculate final signature **s** = (**kSG**+**kRG**, **sS**+**sR**)
6: Calculate public key for **s**: **xG** = **xRG** + **xSG**
7: Verify **s** against excess values in final transaction using **xG**
8: Create Transaction Kernel Containing:
Excess signature **s**
Public excess **xG**
fee
lock_height

End User Experience



dean0h @dean0h 16:06

Hi everyone! I've been wondering if Grin is planning on making transactions more user-friendly at any point in the future. I understand that the use of IP addresses and requirement of both sender and receiver to be online is part of MimbleWimble and essential to maintain a high level of privacy, but will this evolve into a more streamlined approach?
For example, could there ever be a built-in function within the Grin client to send/receive offline transactions more easily?

And not trying to be negative over here, just a suggestion I see as a user. Thank you devs for all the hard work! Very excited to see what the future holds for Grin.

Yeastplume @yeastplume 16:09

@dean0h sure, and I'm taking a screenshot of your quote for the grincon wallet presentation I'm in the middle of putting together. Would you like to say hello to grincon0?

dean0h @dean0h 16:11

@yeastplume Awesome! Yes, definitely fine with that. HELLO grincon0! :D

Yeastplume @yeastplume 16:12

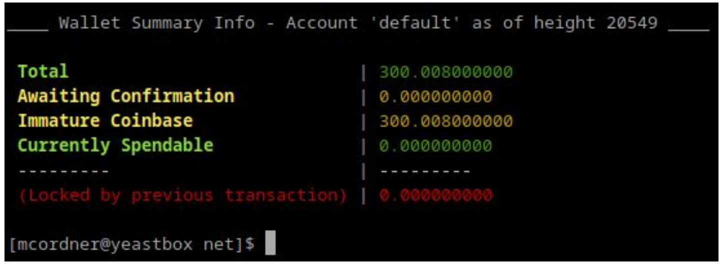
I'll make sure grincon0 says hello back!

Grin's Wallet Strategy

- Don't impose a particular solution
- Provide tools to enable community solutions

Current Grin Wallet

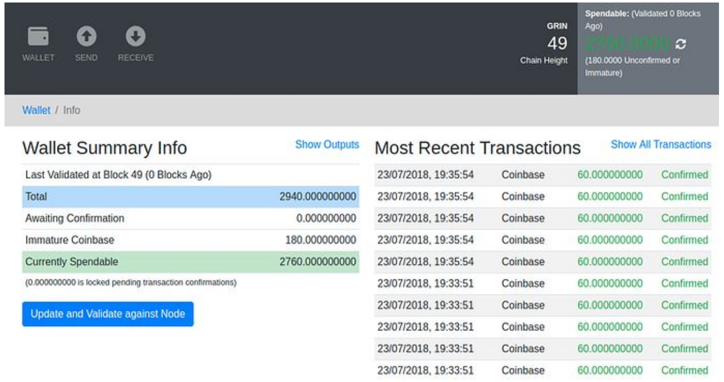
- Command Line
- Web Wallet
- Built from same components



```
____ Wallet Summary Info - Account 'default' as of height 20549 ____

 Total                                 | 300.008000000
 Awaiting Confirmation                 | 0.000000000
 Immature Coinbase                     | 300.008000000
 Currently Spendable                   | 0.000000000
 ---------                             | ---------
 (Locked by previous transaction)      | 0.000000000

[mcordner@yeastbox net]$ 
```


WALLET
SEND
RECEIVE
GRIN
49
Chain Height
Spendable: (Validated 0 Blocks Ago)
2760.0000
(180.0000 Unconfirmed or Immature)
Wallet / Info
Show Outputs
Show All Transactions
Update and Validate against Node

Wallet Summary Info

Last Validated at Block 49 (0 Blocks Ago)	
Total	2940.000000000
Awaiting Confirmation	0.000000000
Immature Coinbase	180.000000000
Currently Spendable	2760.000000000

(0.000000000 is locked pending transaction confirmations)

Most Recent Transactions

23/07/2018, 19:35:54	Coinbase	60.000000000	Confirmed
23/07/2018, 19:35:54	Coinbase	60.000000000	Confirmed
23/07/2018, 19:35:54	Coinbase	60.000000000	Confirmed
23/07/2018, 19:35:54	Coinbase	60.000000000	Confirmed
23/07/2018, 19:35:54	Coinbase	60.000000000	Confirmed
23/07/2018, 19:33:51	Coinbase	60.000000000	Confirmed
23/07/2018, 19:33:51	Coinbase	60.000000000	Confirmed
23/07/2018, 19:33:51	Coinbase	60.000000000	Confirmed
23/07/2018, 19:33:51	Coinbase	60.000000000	Confirmed
23/07/2018, 19:33:51	Coinbase	60.000000000	Confirmed

Current Tx Exchange

- HTTP(S)
    - Anyone interested in receiving needs a wallet listener
    - Works, but very less than ideal (complex, firewalls, NAT traversal)
    - Possibly usable by larger entities

Current Tx Exchange

- File Exchange
  - Slate Sender->Recipient->Sender-> Chain… each step by emailing/schlepping file around
  - Cumbersome (especially for > 2 transactors)
  - State needs to be kept around indefinitely in everyone’s wallet until confirmed
  - “Safest” launch approach

Tx Exchange

- Other ideas
  - Semi-trusted 3rd party service / relay
  - i2p
  - Distributed Hash Table for user lookup

Grin Wallet Structure / Components

- grin_libsecp256k1
  - Elements project + aggsig
- grin_keychain
  - BIP-32 keychain implementation
- libtx (within grin_wallet crate)
  - Low level transaction building, aggsig, proofs etc
  - Built on our fork of rust-libsecp256k1

Grin Wallet Structure / Components

- libwallet (within grin_wallet crate)
    - Bit higher up -
    - libwallet/internal - output selection, wallet content updating, wallet restore, etc
    - APIs on which to build wallet clients
        - HTTP Listener
        - In-process
- grin_wallet crate
    - Full wallet implementation with LMDB backend (implements libwallet traits)

Community Projects

- Grinbox Relay Service
  - http://grinbox.io/
  - Non-trusted service to route messages
- Blockcypher - Wallet

Q+A

https://grin-tech.org