Private Key Management

Manage your keys in Polkadot

David Dorgan
Devops @ Parity Technologies Ltd.

david@parity.io

PoW vs PoS Key Management

- Keys are a very static thing in PoW systems
- Minimal risk in PoW (except spent resources!)
- A totally different game in PoS (or NPoS)
- Non-malicious behaviour can lead to small slashing conditions (e.g. unavailability)
- Slashing on a sliding scale from .1% to 100%

Polkadot key types

- Account Keys

  Controller & Stash Keys

- Session keys

  Grandpa / Babe / I’m Online / Parachain

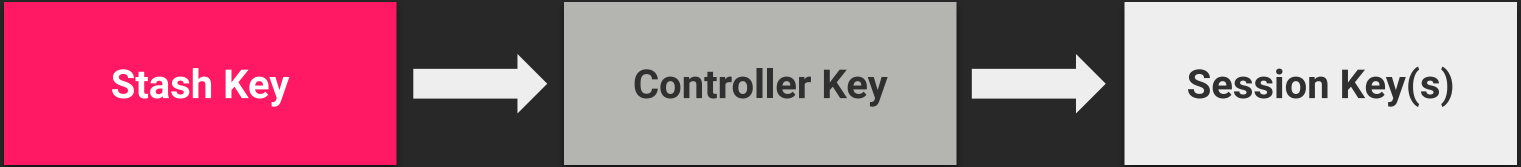
- Transport layer keys

  For the libp2p networking layer

Polkadot key management

- Has been changed recently
- Avoid exposing these via the cli
- Encourage the use of keystores and a single aggregate key that can be used with set_sessionKey

Key flow



Cold; Sr25519

Stash Key

Bonding
Transfer funds

Warm; Sr25519

Controller Key

Set preferences
Set staking mode

Hot; sr25519 / ed25519, BLS

Session Key(s)

Author blocks
Attest to valid parachains
Finalise chains
Decrypt validator IPs

Account Keys

- The **controller** key is a semi-online key that will be in the direct control of a user, and used to submit manual extrinsics. It should have some funds but just enough for basic operations.
- Since the **stash** key is kept offline, it must be set to have its funds bonded to a particular controller.

Session Keys

- BABE: sr25519 - Block authoring
- GRANDPA: ed25519 - Block finalization
- I'm Online: sr25519 - Sent on each session by validators
- Parachain: sr25519 - For signing parachains blocks

Transport Layer Keys

- Can be provided as a seed on startup (--node-key

  0x5a44bf0bac9438b3b02d6ac9b09a739cb9c0029400f3b7d07be74edea7c42f4d)
- Essential on bootnodes but relevant for all nodes.
- Produces a deterministic multiaddr

  E.g. /ip4/10.12.4.209/tcp/30333/p2p/QmSk5HQbn6LhUwDiNMseVUjuRYhEtYj4aUZ6WfWoGURpdV

Key Creation

- **Subkey** is the tool of choice for key creation and management
- Useful options:
  - generate - create a random key and display the Phrase / Seed / Public Key / SS58 Address
  - inspect - Show the Phrase / Seed / Public Key / SS58 Address for a known input (phrase or key).
  - vanity - Generate a seed that provides a vanity address

Key Creation - continued

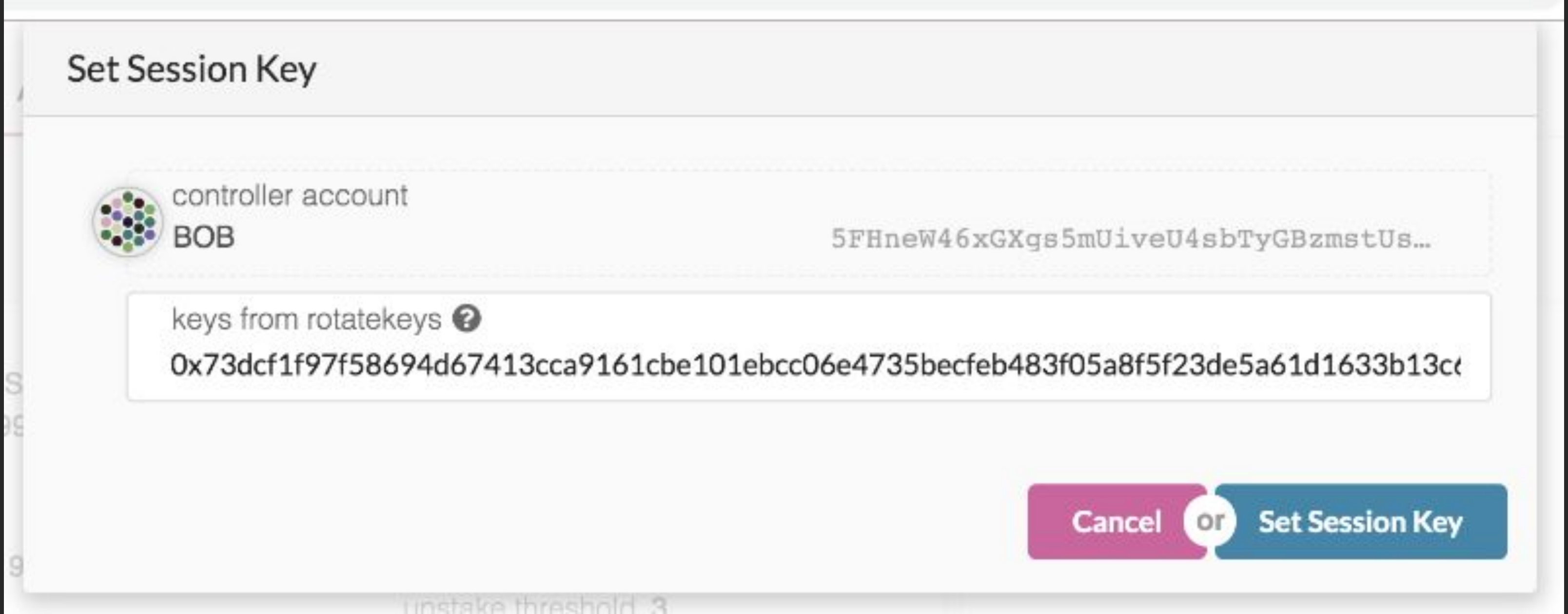
- **Key Types**
    - subkey --ed25519 generate - Create an Edwards 25519 key
    - subkey --sr25519 generate - Create a Schnorr/Ristretto x25519 key

Keystore for validators - The easy way

- Create a keystore via the **author_rotateKeys**
- Will return four public keys and save the private keys in your keystore file. This can also be used “**Set Session**” on the UI!

```
$ curl -H "Content-Type: application/json" --data '{ "jsonrpc":"2.0", "method":"author_rotateKeys", "params":[],"id":1 }'
localhost:9933
{"jsonrpc":"2.0","result":"0x9e8c8add4a6351759e7d3f574633979ac65ab01f2f2b09b02c3e501731e240a760542d7faf
bbe69c34e80233d01a64ecb8ff4a091364ad4e6ed102ad6a391875ec59cbe92b8b5bd856df986634581b7d04d57a18
988ddaa772214f3cb45a8c48","id":1}
```

Keystore for validators - The easy way



Set Session Key

controller account
BOB
5FHneW46xGXgs5mUiveU4sbTyGBzmstUs...

keys from rotatekeys
0x73dcf1f97f58694d67413cca9161cbe101ebcc06e4735becfeb483f05a8f5f23de5a61d1633b13c

Keystore for validators - Using specific keys

- --key has been replaced with four new keys
- gran - grandpa Key / babe - babe key / imon - I’m online key / para - parachain key

```
KEY_SEED="xxxxx"
RPC_ENDPOINT=localhost:9933
for KEY_TYPE in $(echo gran babe imon para ) ; do
             curl -H "Content-Type: application/json" \
            --data '{ "jsonrpc":"2.0", "method":"author_insertKey", "params":["'"${KEY_TYPE}"'", "'"${KEY_SEED}"'"],"id":1 }' \
            "${RPC_ENDPOINT}"
done
```

parity

Subkey Demo Time!

Subkey Demo

We’re hiring!

parity.io/jobs

Parity updates and events

parity.io/newsletter

Questions?

david@parity.io