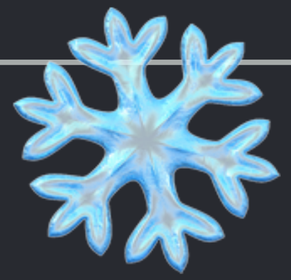
BLOCKCHAIN COMMONS

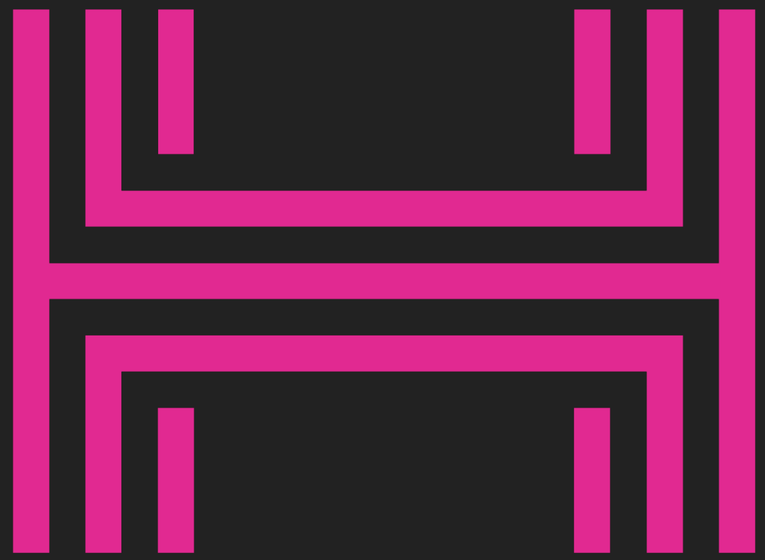
LEARNING FROST



WHAT IS BLOCKCHAIN COMMONS?

- We are a community that brings together stakeholders to collaboratively build open & interoperable, secure & compassionate infrastructure.
- We design decentralized solutions where everyone wins.
- We are a neutral “not-for-profit” that enables people to control their own digital destiny.

THANKS TO OUR FROST SPONSOR



Human
Rights
Foundation

AN OVERVIEW

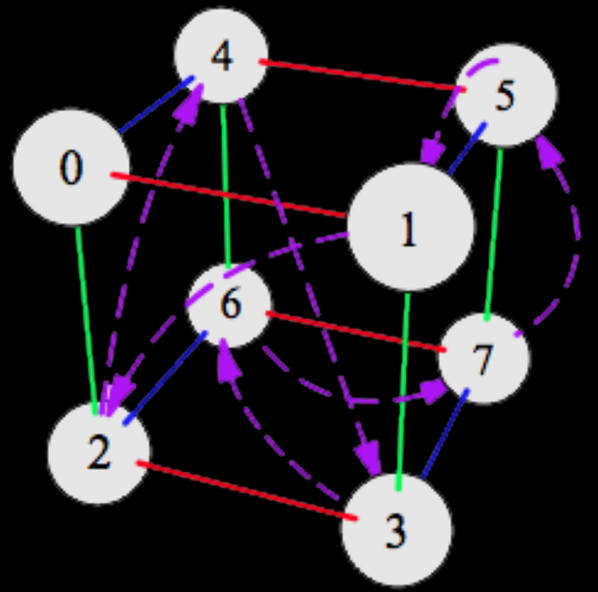
WHAT IS FROST?

WHAT IS FROST?

Flexible **R**ound-**O**ptimized **S**chnorr **T**hreshold

It’s a:

- **Threshold Signature** Scheme
- Using the **Schnorr** algorithm
- Built on Discrete Logs over Finite Fields
- With a specific methodology for signing
- That works with **Distributed Key Generation**



ABOUT SIGNATURES

Digital signatures are used to verify messages:

- **Private** key signs a message
- **Public** key verifies that signature

You know that:

- A specific person authorized the message
- The message is not changed

Signatures are not the same as encryption.

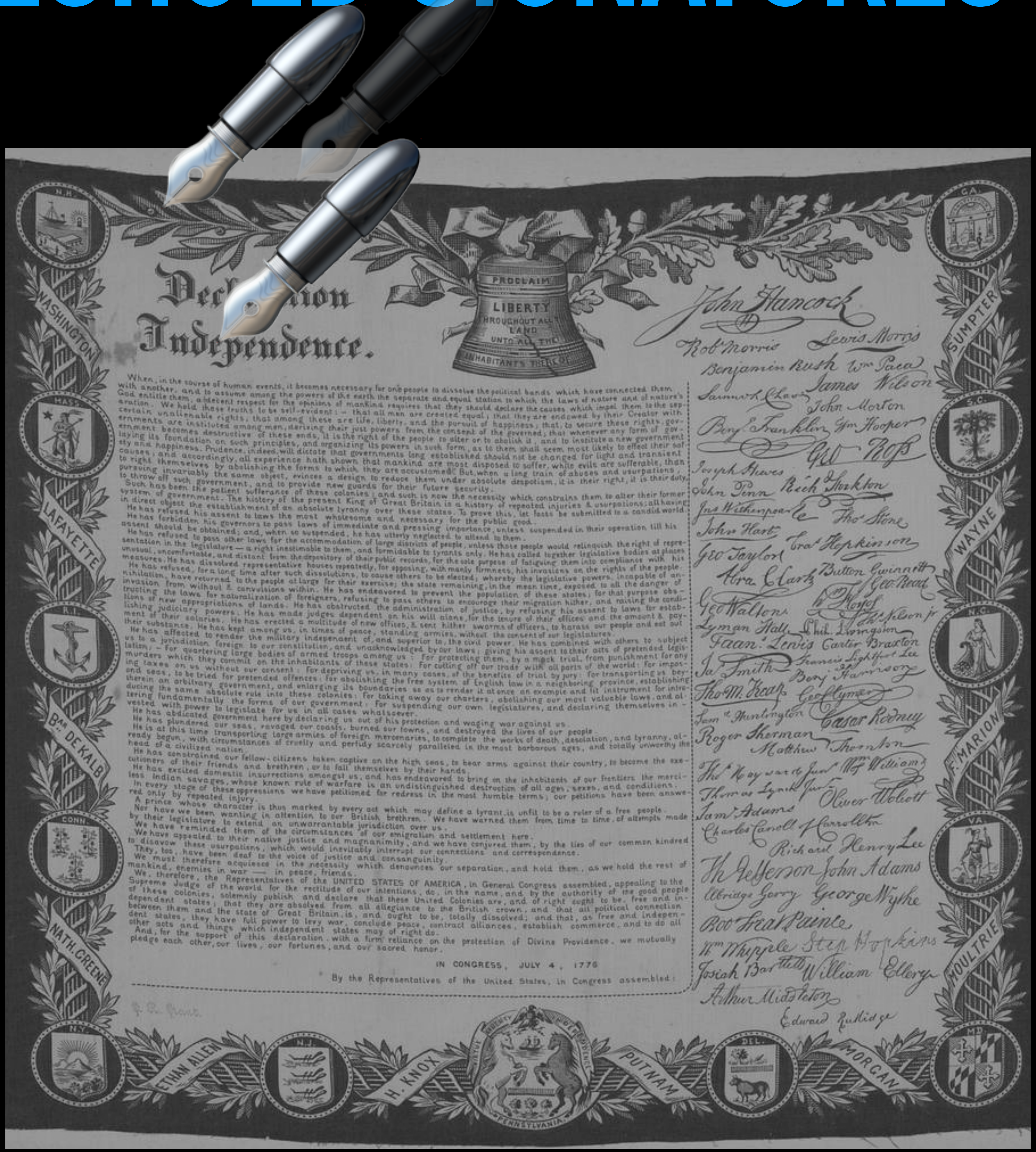
ABOUT MULTISIGS & THRESHOLD SIGNATURES

Signatures can involve:

- A group of **signers** (multisig)
- A lower **threshold** for how many signers are required for authorization (threshold sig)

These are often defined as:

- **m-of-n**: a subset (m) of the group (n) may sign
- **n-of-n**: all (n) of the group (n) must sign



<u>FROST is an m-of-n multisig system</u>

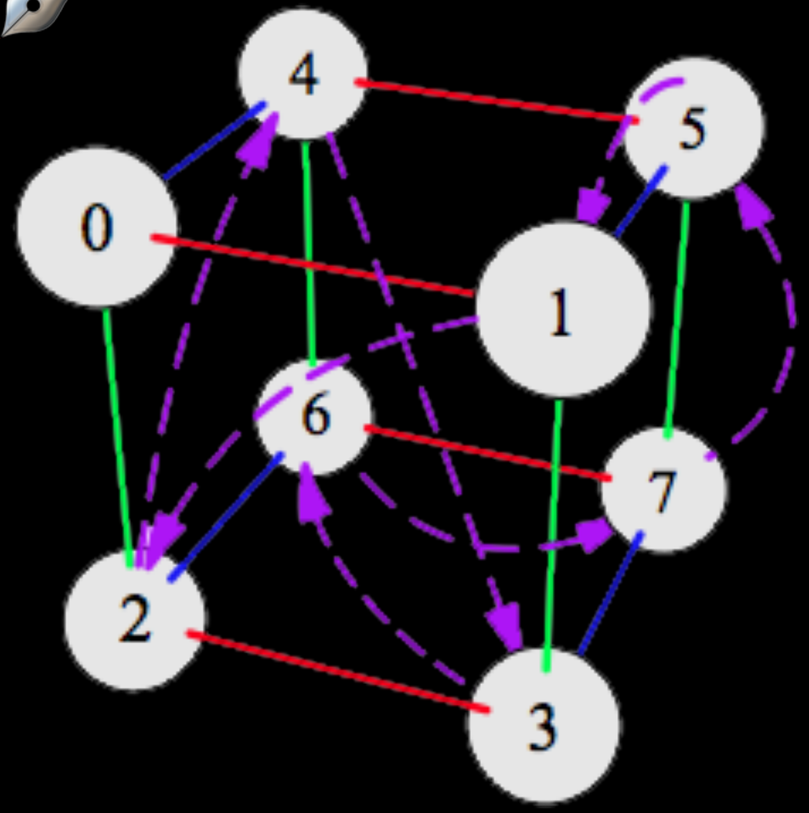
ABOUT SCHNORR SIGNATURES

Schnorr is a particular signature algorithm

- Focused on discrete logs & finite fields
- But that's not relevant to understanding it.

Its big advantage?

- Signatures are **aggregatable**
- Can be added together
- But always the **same size**



ABOUT DISTRIBUTED KEY GENERATION (DKG)

DKG is a Multiparty Computer (**MPC**) method

- Multiple computers together create a key
- The key never exists in one place!
- Each user just has a **share** (a fragment)

DKG is one method for FROST key generation

- Trusted Dealer Generation is the other
- But DKG is much more secure

ADVANTAGES OVER OTHER MULTISIG SYSTEMS

WHY USE FROST?

FROST VS BITCOIN MULTISIG

Classic **Bitcoin** Multisig

- Pay to Script Hash (P2SH)
- OP_CHECKMULTISIG
- Explicitly says it’s a multisig.
- The signature can be long

FROST VS BITCOIN MULTISIG

New **FROST** Signature

- Pay to Taproot (P2TR)
- Just a signature!
  - *(and maybe a Merkle Tree hash)*
- Can’t tell it’s a multisig
- Can’t tell how many people signed
- Can’t tell who signed

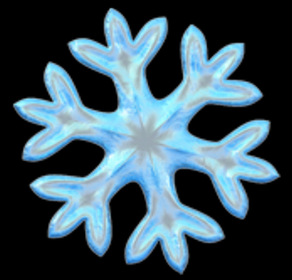
FROST VS MUSIG2

MuSig2 is Another Schnorr Signature Scheme

- Only n-of-n threshold (natively)
- Accountable: you know who signed

vs **FROST**:

- m-of-n threshold
- Deniable: you don’t know who signed



MuSig2

MULTISIGNATURE COMPARISONS

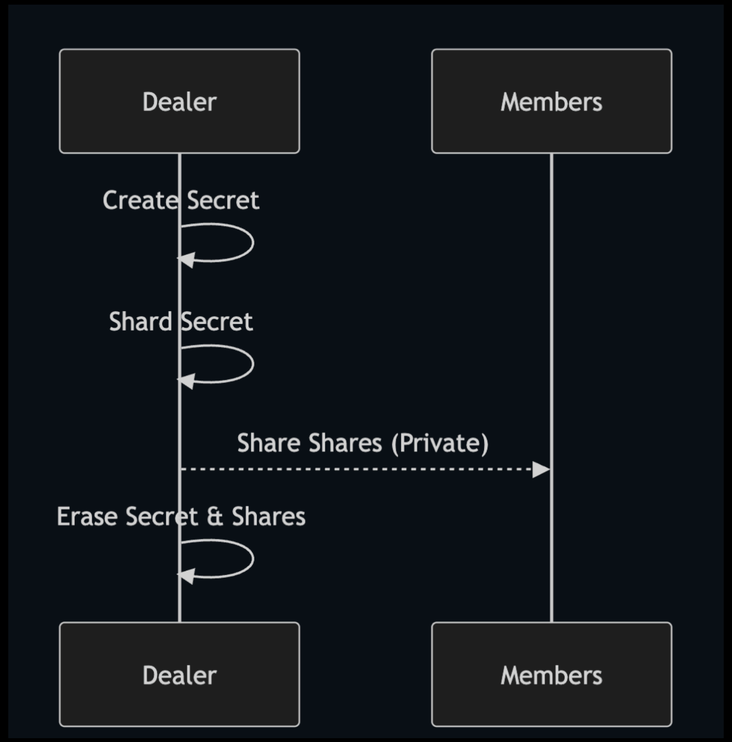
	FROST	MuSig2	Multisig
Scheme	Schnorr	Schnorr	ECDSA
Threshold	m-of-n	n-of-n	m-of-n
Privacy	Deniable	Accountable	Accountable
Signing	2 Rounds or Preprocess	2 Rounds	1 Serial Round
Size	64 bytes	64 bytes	72 bytes/sig

THE ADVANTAGES OF FROST (IN SUMMARY)

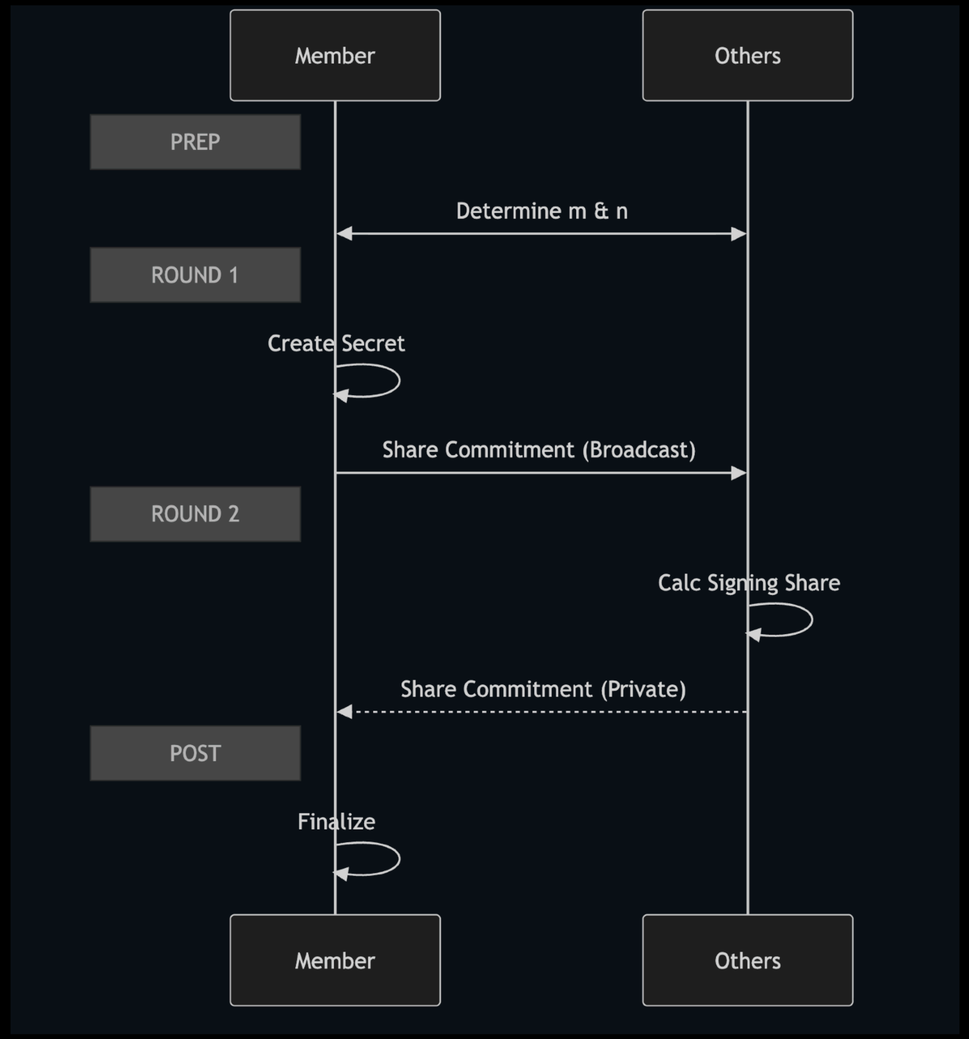
- Small Signatures
- Private Signatures
- Efficient Communication
- Strong Security (with DKG)
- Refresh & Repair Capabilities

A QUICK LOOK AT FROST PROCESSES

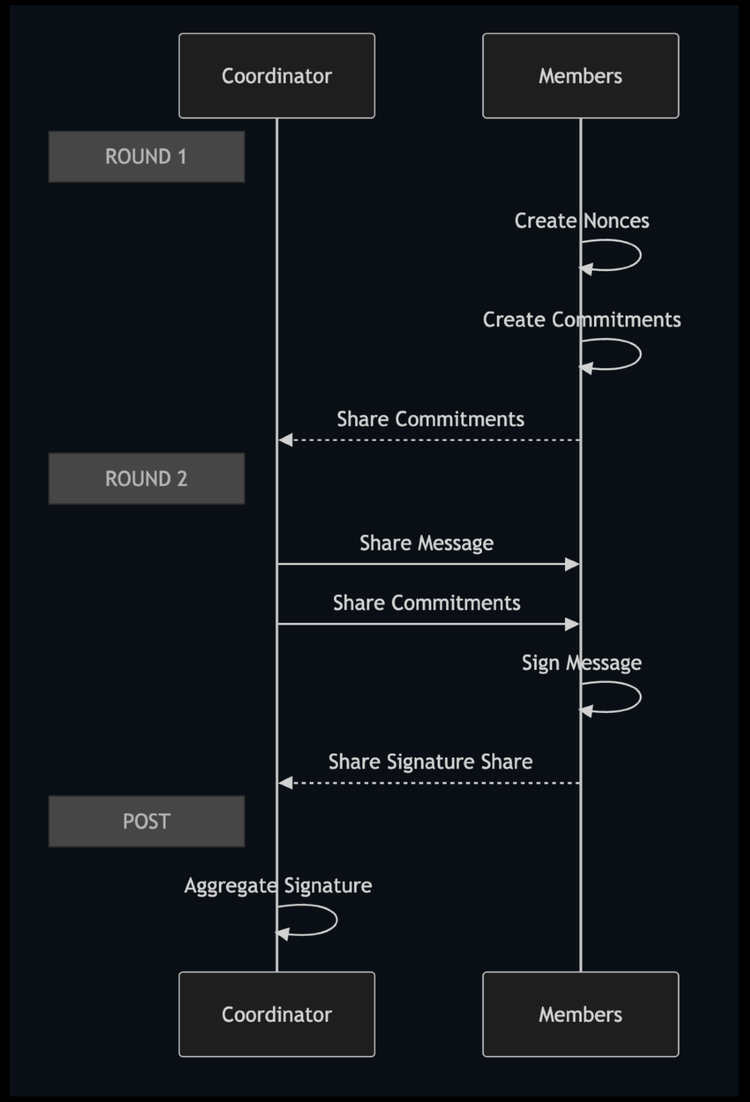
HOW DOES FROST WORK?



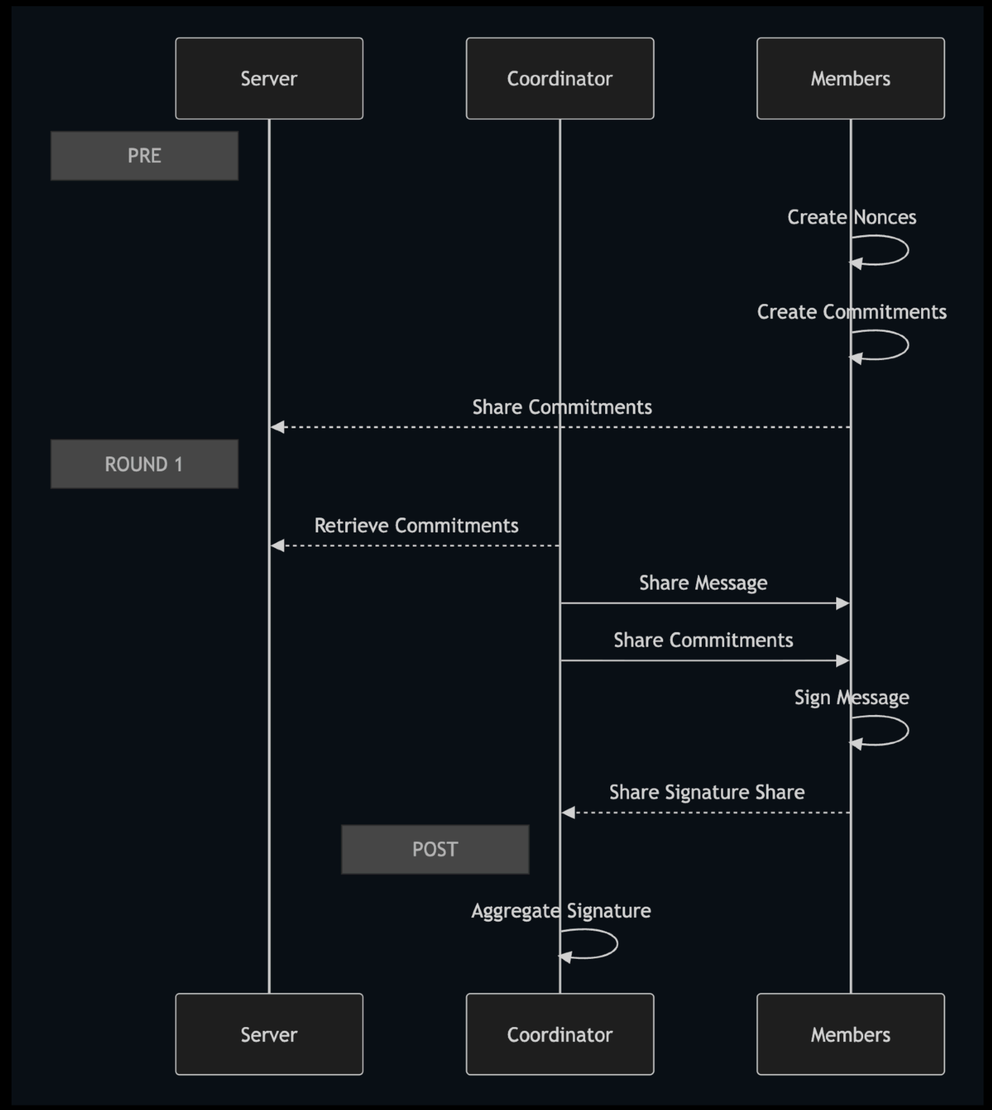
Key Generation: Trusted Dealer



Key Generation: Distributed Key Generation (DKG)



Signing: Two Rounds



Signing: One Round with Preprocessing

A NEW BLOCKCHAIN COMMONS COURSE

LEARNING FROST FROM THE COMMAND LINE

ABOUT THE COURSE

We have released the *Learning FROST from the Command Line* course.

- Written in the style of …
  - *Learning Bitcoin from the Command Line*
- Sponsored by HRF
- The start of this presentation was Chapter 1

https://learningfrost.blockchaincommons.com/

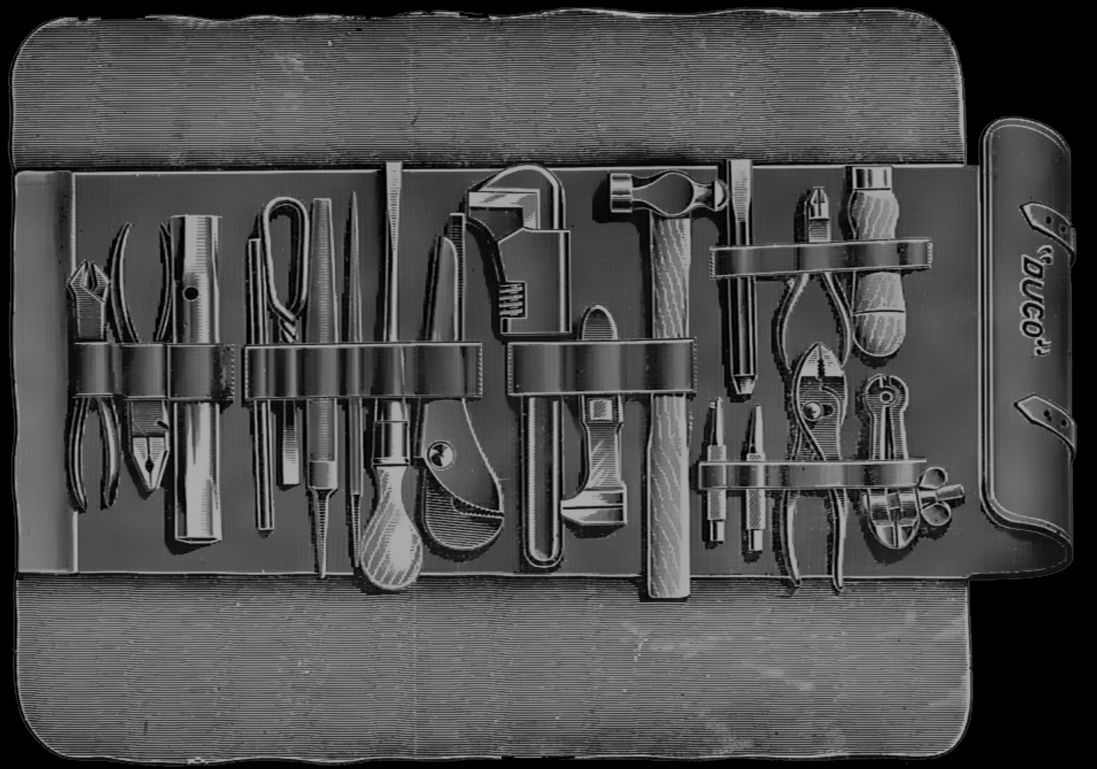
ABOUT THE FROST TOOLS

Course built on ZF FROST Tools:

- trusted-dealer
- dkg
- coordinator
- participant
- frostd server
- frost-client

Installable with **Cargo**.

https://github.com/ZcashFoundation/frost-tools



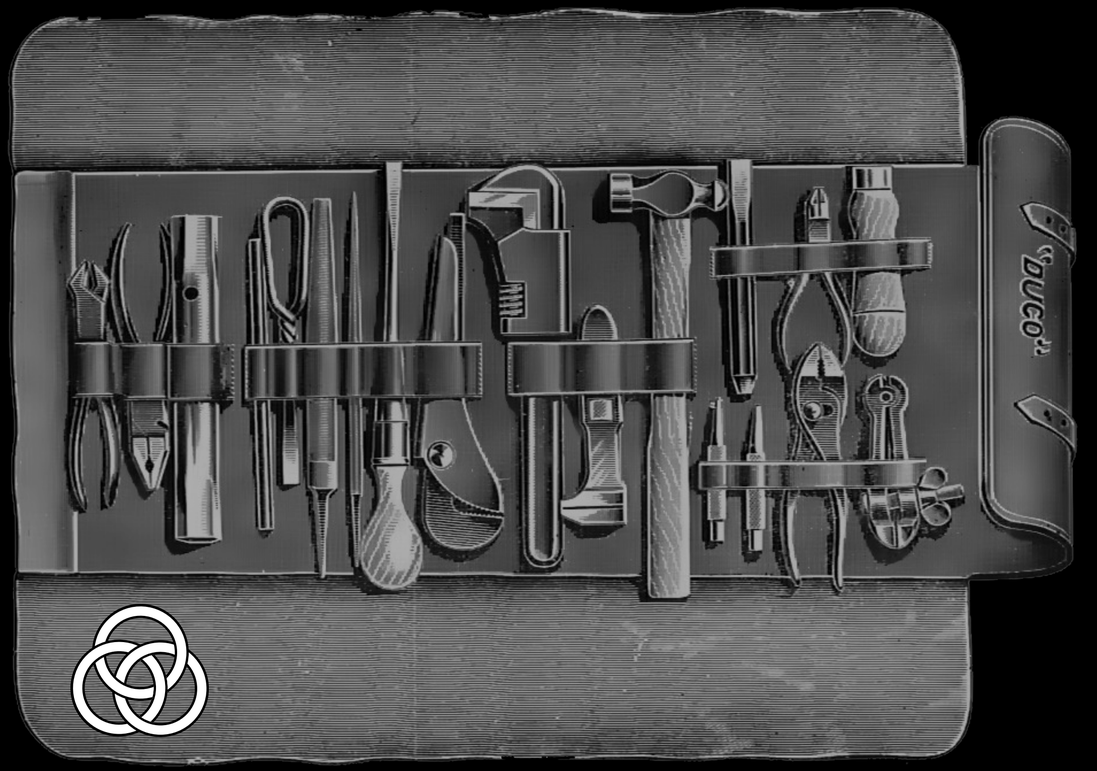
OUR EXPANSION TO THE FROST TOOLS

frost-verify-rust

- A tool for checking ZF FROST Signatures

Tools for **Signing Bitcoin transactions**

- Secp256K1-TR Ciphersuite for ZF FROST
- Taproot Tweak for ZF FROST
- Helper to Extract Sighash from PSBT
- Helper to Insert Signature into PSBT



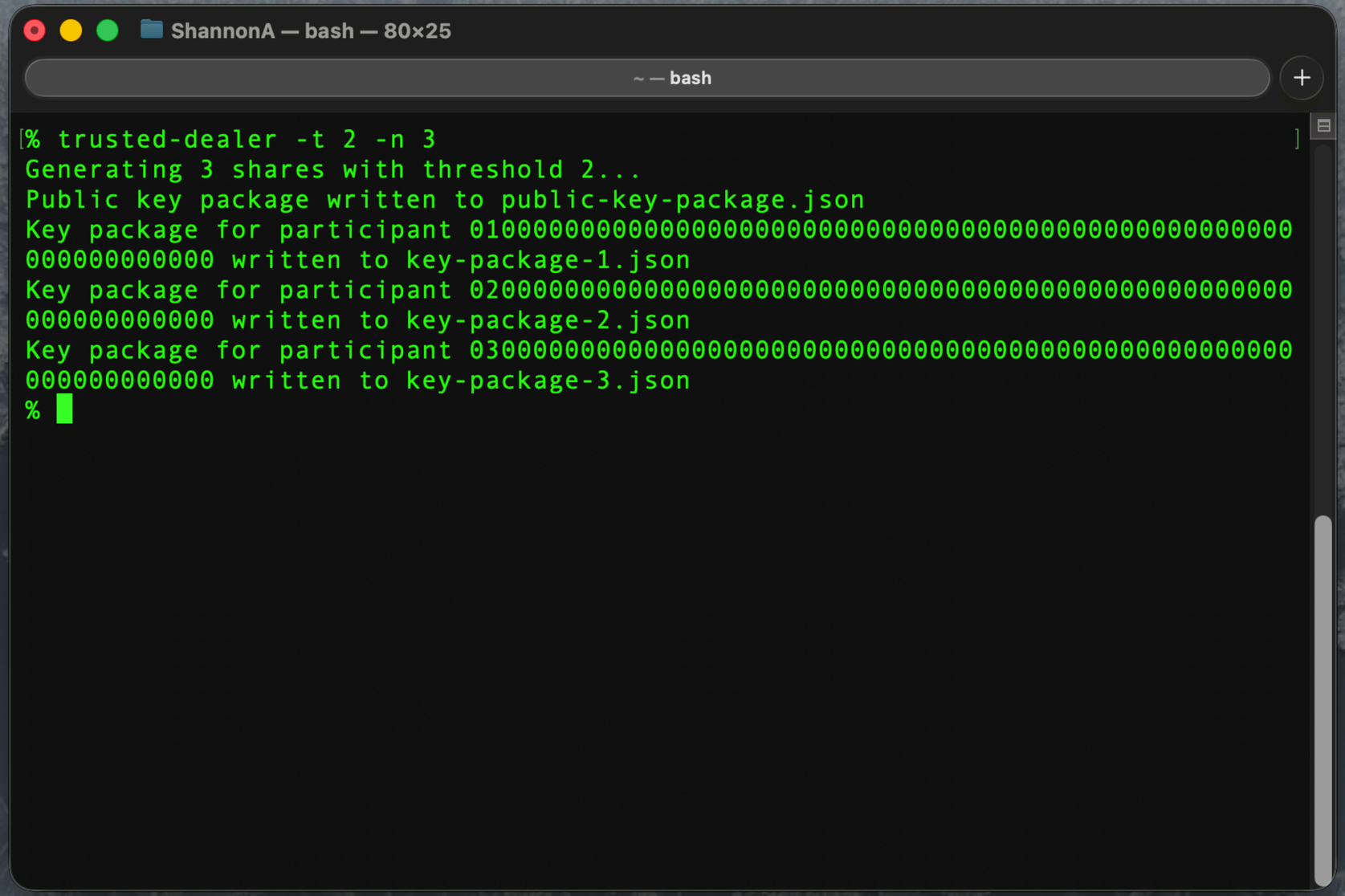
A QUICK OVERVIEW

The course offers the hands-on experience!

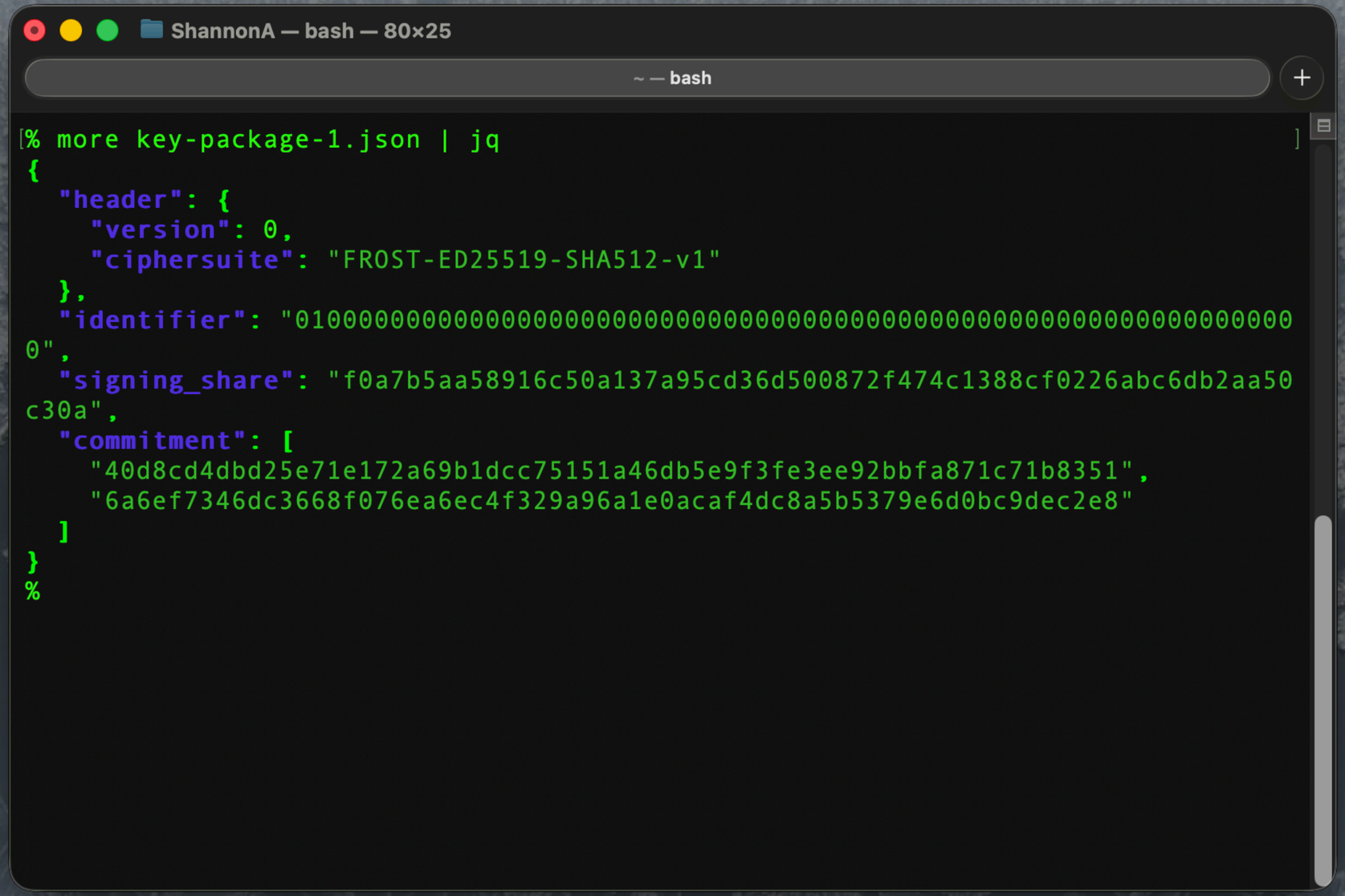
- That’s how you learn how stuff works
- But here’s a quick look at what it contains

Also take a look at Wolf’s two Bitcoin videos

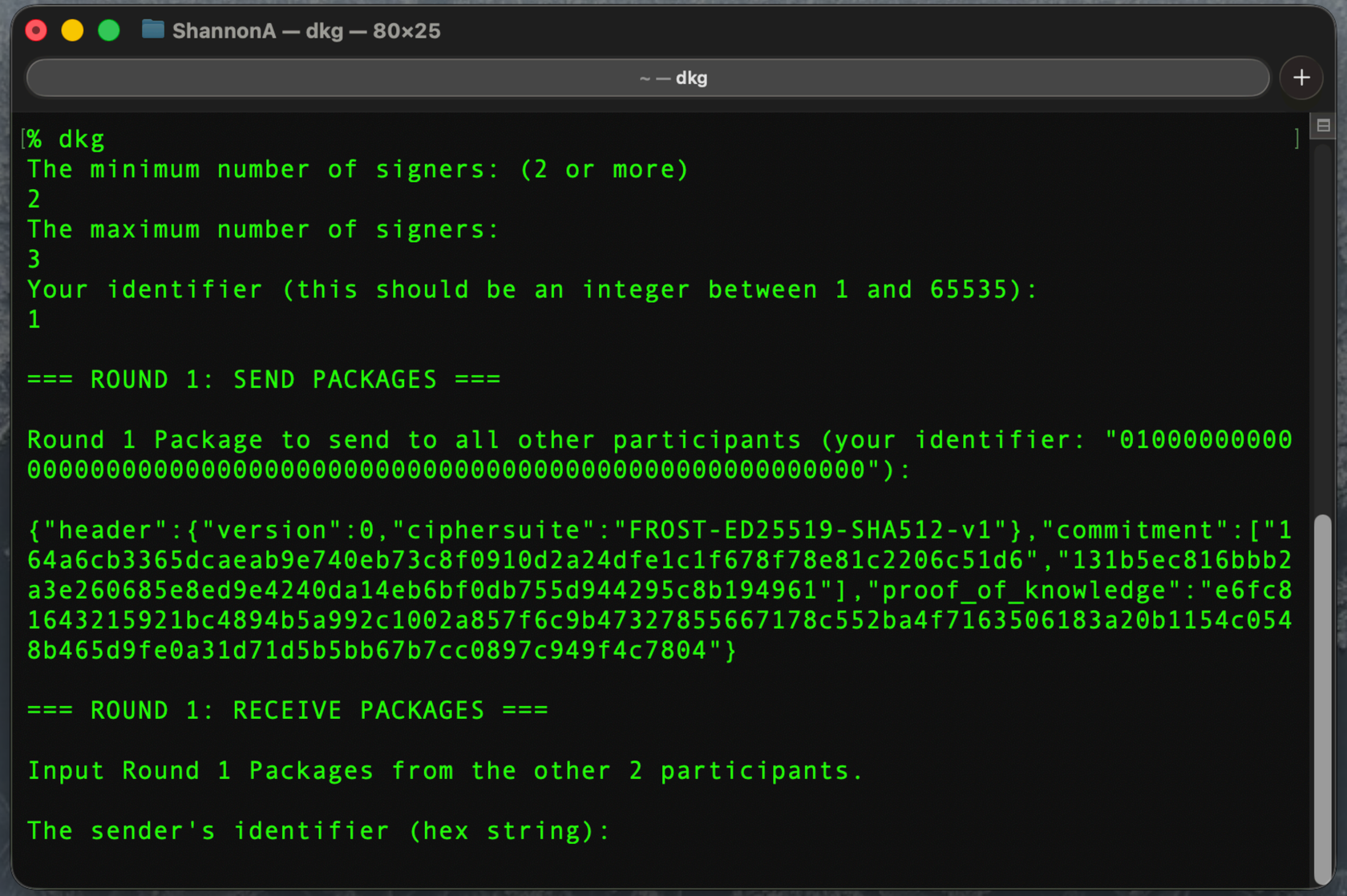
- FROST-CLI Demo Meeting
- FROST DKG Bitcoin Signing Demo



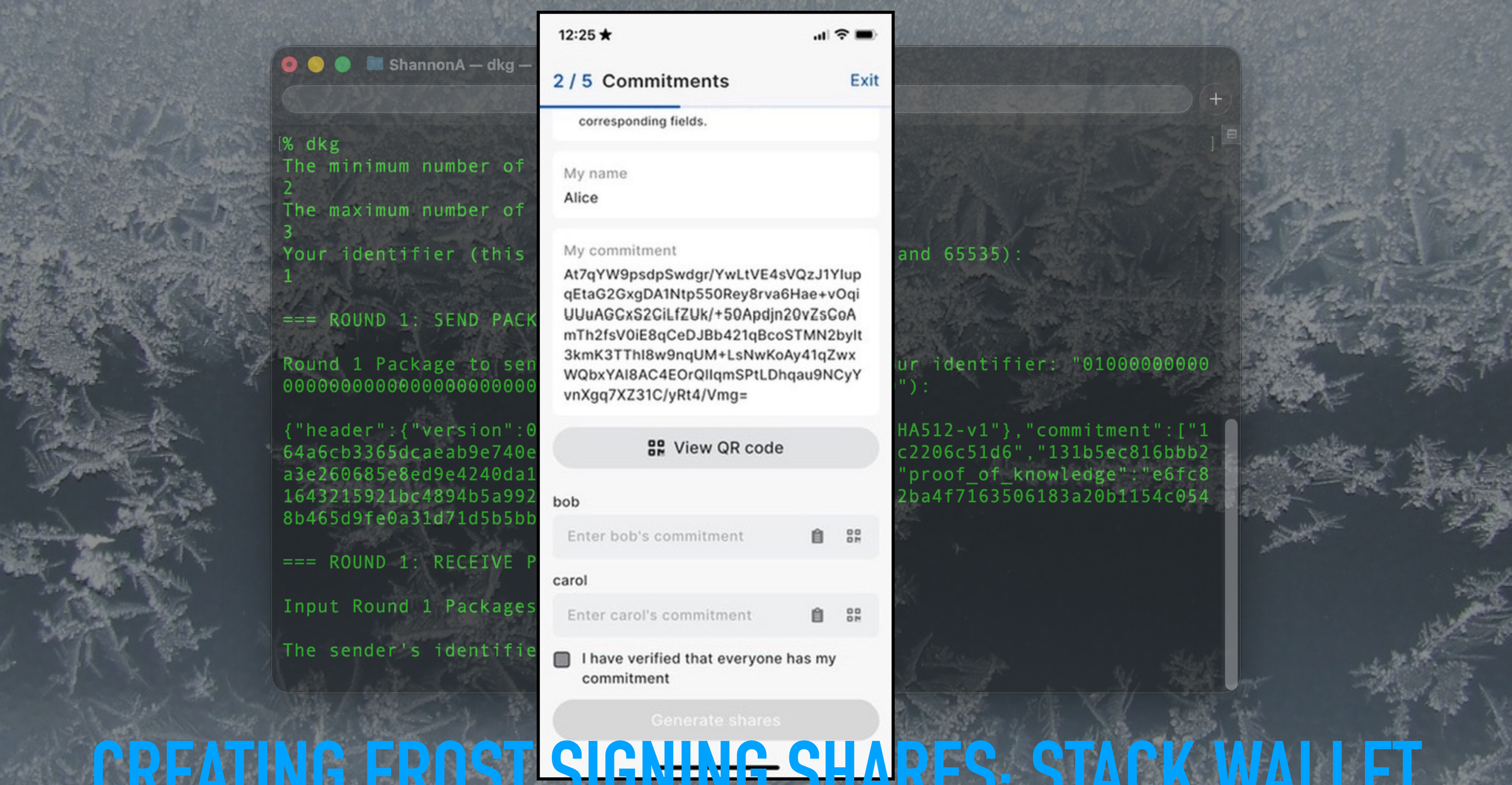
CREATING FROST SIGNING SHARES: TRUSTED DEALER



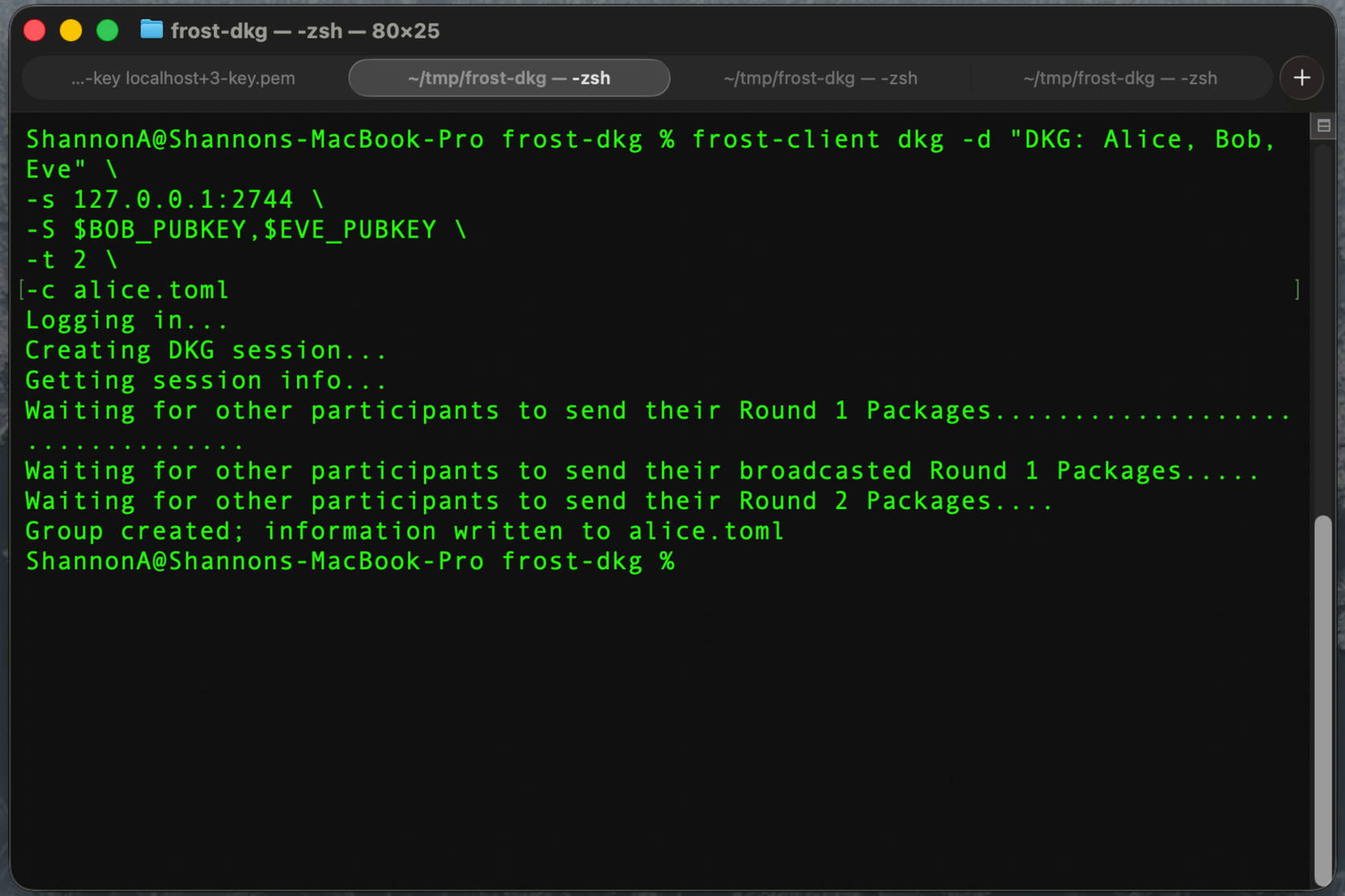
CREATING FROST SIGNING SHARES: TRUSTED DEALER



CREATING FROST SIGNING SHARES: DKG



CREATING FROST SIGNING SHARES: STACK WALLET



CREATING FROST SIGNING SHARES: DKG WITH SERVER

FROST FACES TYPICAL KEY MANAGEMENT ISSUES

You now have a file with your signing share!

- How do you secure your signing share?
- How do you record its intended usage?
- How do you keep it from being lost?

But **FROST** has big advantages due to thresholds:

- if m > 1, one stolen key doesn’t steal signature
- if m < n, one lost key doesn’t lose signature

frost-dkg — -zsh — 80×25

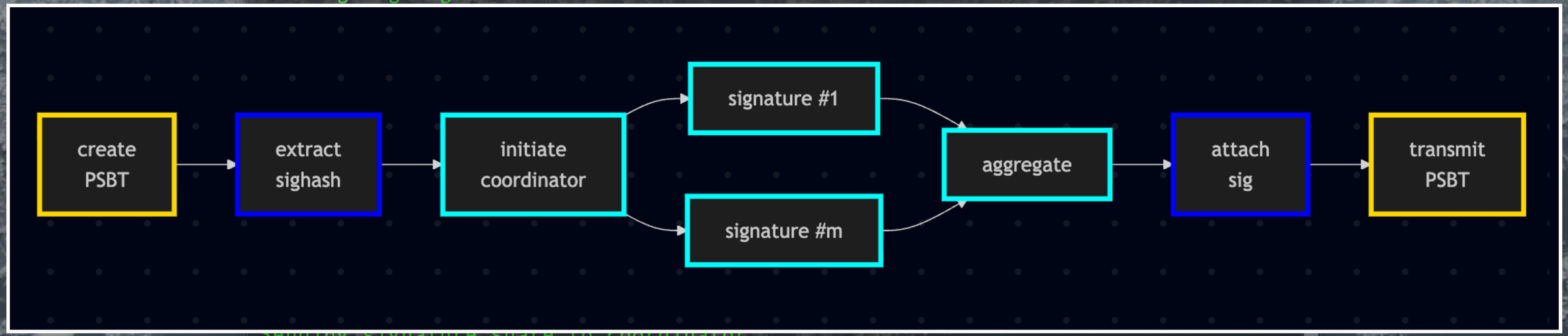
...-key localhost+3-key.pem | ...g-250917.sig -c alice.toml | ...7.0.1:2744 -c alice.toml | ~/tmp/frost-dkg — -zsh

```
ShannonA@Shannons-MacBook-Pro frost-dkg % GROUP_ID=$(grep -oE '^\[group\.[0-9a-f
]+' bob.toml | head -1 | sed 's/^\[group\.//')
frost-client participant -g $GROUP_ID -s 127.0.0.1:2744 -c bob.toml
Logging in...
Joining signing session...
Sending commitments to coordinator...
Waiting for coordinator to send signing package.....
Signing package received
Message to be signed (hex-encoded):
446174653a20392f31372f32350a50726573656e743a2030313030303030303030303030303030
30303030303030303030303030303030303030303030303030303030303030303030303030303030
303030303030302c20303230303030303030303030303030303030303030303030303030303030
303030303030303030303030303030303030303030303030303030303030303030300a0a4465636973
696f6e733a0a2a20416c6c6f636174652024313020746f204a616d20696e2043656e7472616c2050
61726b20506172747900a2a20416c6c204d656d62657273204d75737420436f6e74726962757465 20
243520746f2047726f7570204b69747479206279203920392f32342f32350a
Do you want to sign it? (y/n)
y
Sending signature share to coordinator...
Done
ShannonA@Shannons-MacBook-Pro frost-dkg %
ShannonA@Shannons-MacBook-Pro frost-dkg %
ShannonA@Shannons-MacBook-Pro frost-dkg %
ShannonA@Shannons-MacBook-Pro frost-dkg %
ShannonA@Shannons-MacBook-Pro frost-dkg %
```

SIGNING WITH FROST

```
frost-dkg — -zsh — 80×25
...-key localhost+3-key.pem   ...g-250917.sig -c alice.toml   ...7.0.1:2744 -c alice.toml   ~/tmp/frost-dkg — -zsh

ShannonA@Shannons-MacBook-Pro frost-dkg % GROUP_ID=$(grep -oE '^\[group\.[0-9a-f
]+' bob.toml | head -1 | sed 's/^\[group\.//')
[frost-client participant -g $GROUP_ID -s 127.0.0.1:2744 -c bob.toml
Logging in...
Joining signing session...
Sending signature share to coordinator...
Done
[ShannonA@Shannons-MacBook-Pro frost-dkg %
[ShannonA@Shannons-MacBook-Pro frost-dkg %
[ShannonA@Shannons-MacBook-Pro frost-dkg %
[ShannonA@Shannons-MacBook-Pro frost-dkg %
ShannonA@Shannons-MacBook-Pro frost-dkg %
```



SIGNING BITCOIN WITH FROST

THAT'S FROST IN A NUTSHELL

This is a capstone of two years' work.

- We've held five **FROST** meetings
- For implementers & developers

We're now moving from discussion to implementation:

- Learning **FROST** is just the start
- We are also bringing **FROST** into our stack

Our ultimate goal?

- Wider usage of **FROST**!

https://developer.blockchaincommons.com/frost/

WHAT'S NEXT?

FUTURE WORK

SIGNING EXAMPLES

So what do you sign with FROST?

- Gordian Envelope
- Gordian Club Updates

COORDINATION EXAMPLES

ZF FROST offers two types of coordination

- By Hand (ugh!)
- By Coordinator (centralized!)

How do we get advantages of coordination without centralization?

- Hubert, the Dead-Drop Hub

https://developer.blockchaincommons.com/hubert/


ANY QUESTIONS?

Learning FROST from the Command Line

https://learningfrost.blockchaincommons.com

Shannon Appelcline

shannon.appelcline@gmail.com

@BlockchainComns

"Advocating for the creation of open, interoperable, secure & compassionate digital infrastructure to enable people to control their own digital destiny and to maintain their human dignity online"