

Time-locked Recovery Factors

A thought experiment in key recovery for MPC

Leonard Tan

CTO, Web3Auth



What's the problem?

Seed phrases bad

Seed phrases bad: use MPC / multisigs pls

Seed phrases contribute to loss, theft, and terrorism.

- Seed phrases are like a very long password
 - Easy to lose
 - Easy to compromise

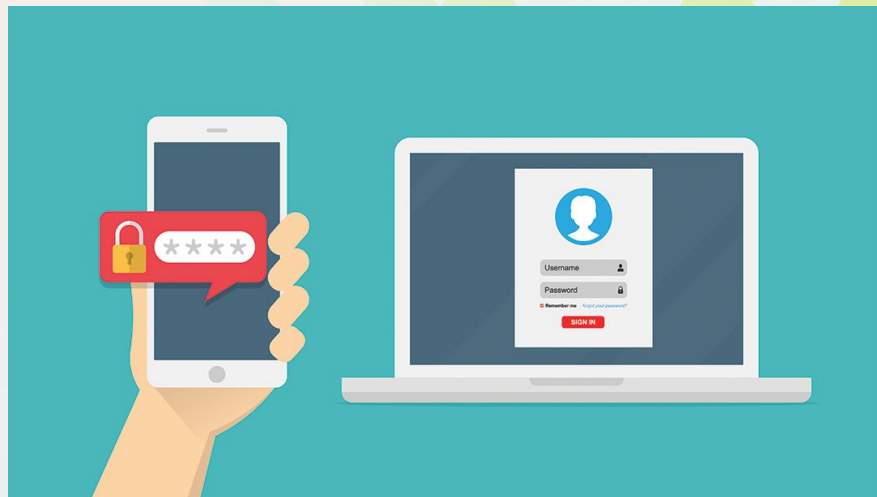


You, basically, when you write down a seed phrase

Seed phrases bad: use MPC / multisigs pls

We should be using multi-factor authentication

- It is the outcome of decades of evolution in user authentication (circa. 1986)
- It works!
 - Provides better security
 - Prevents phishing attacks
 - Relatively easy to set up



You, hopefully, after this talk is over

Seed phrases bad: use MPC / multisigs pls

MPC and multisigs have very similar properties:

- Better than a seed phrase
- More hack resistant
- Better redundancy
- Additional checks

But there are some differences:

- Multisigs can support more complex access structures (i.e. organisations)
- MPC costs less to deploy and operate (off-chain is cheaper, no gas)
- Multisigs have trustless access to on-chain state (prices & limits w/o oracles)
- MPC is cross-chain (same setup, works across different L1s and L2s)

Seed phrases bad: use MPC + multisigs pls

MPC and multisigs

- Better than seed phrases
- Multi-factor authentication
- Additional security

But there are some downsides

- Multisigs can be slow
- MPC costs more
- Multisigs have a learning curve
- MPC is cross-chain



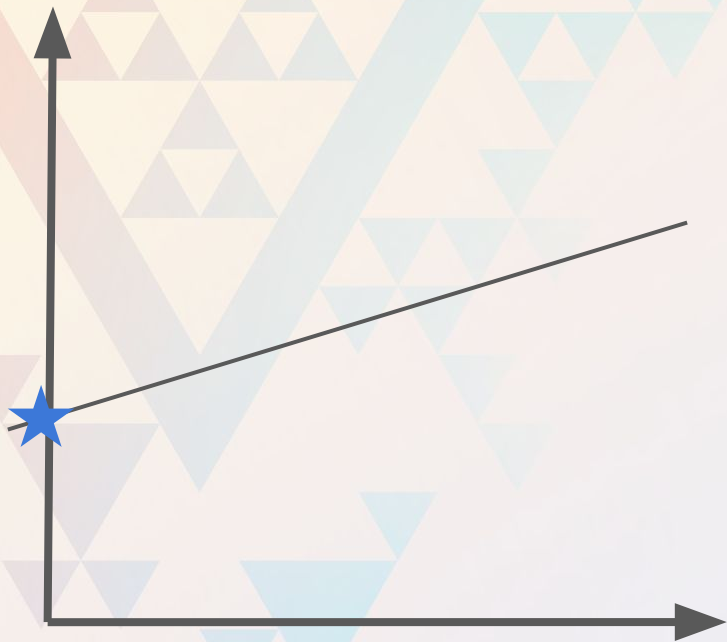
WHY NOT BOTH?

(organisations)
(no gas)
(units w/o oracles)
(and L2s)



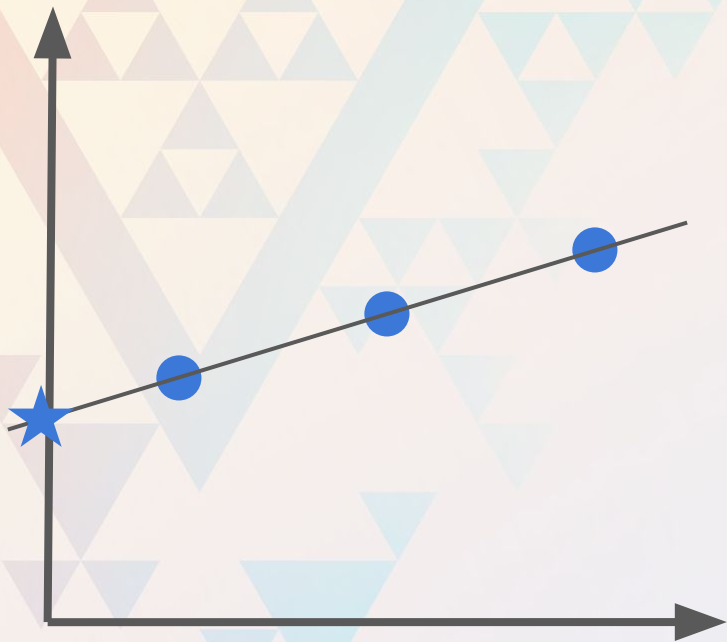
MPC?

What is MPC?



First, we need to understand how to share a secret key

Imagine you drew a line, and the y-intercept (star) represents your private key.



First, we need to understand how to share a secret key

Imagine you drew a line, and the y-intercept (star) represents your private key.

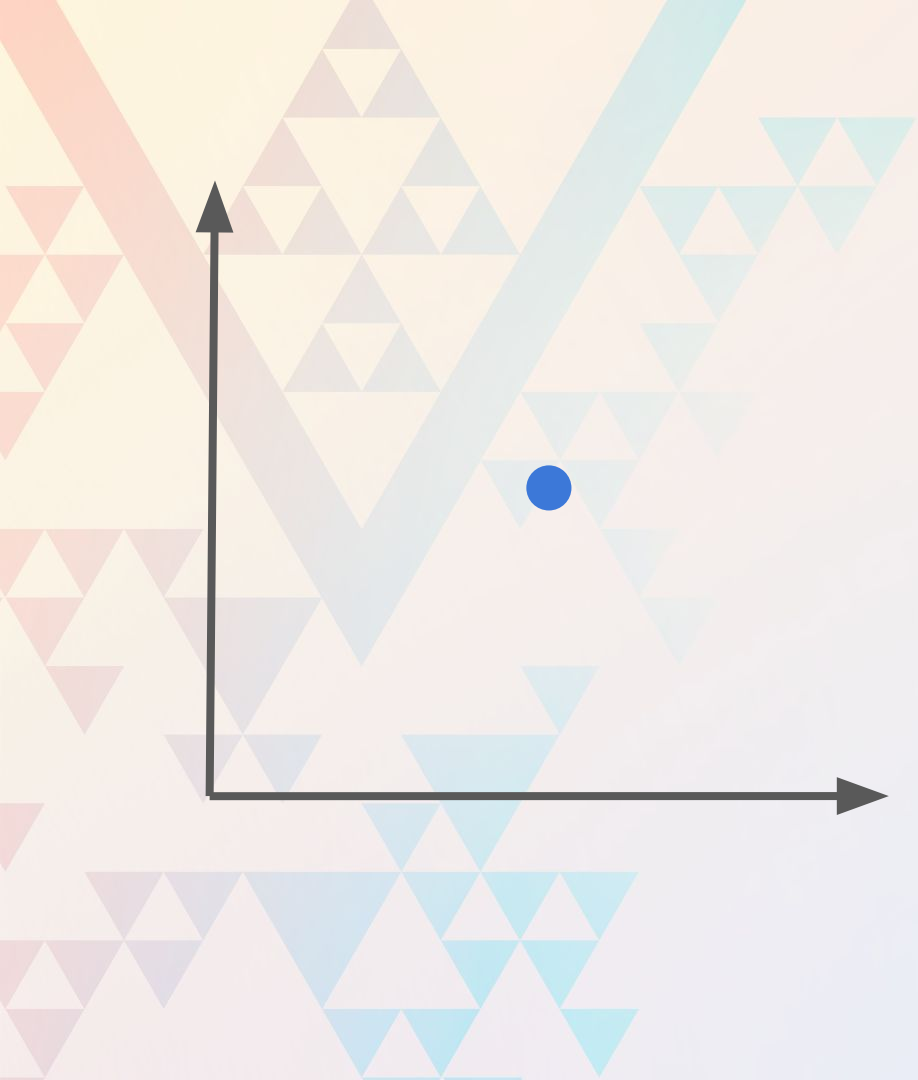
Now you draw 3 points on the line...



First, we need to understand how to share a secret key

Imagine you drew a line, and the y-intercept (star) represents your private key.

Now you draw 3 points on the line... and erase the line!

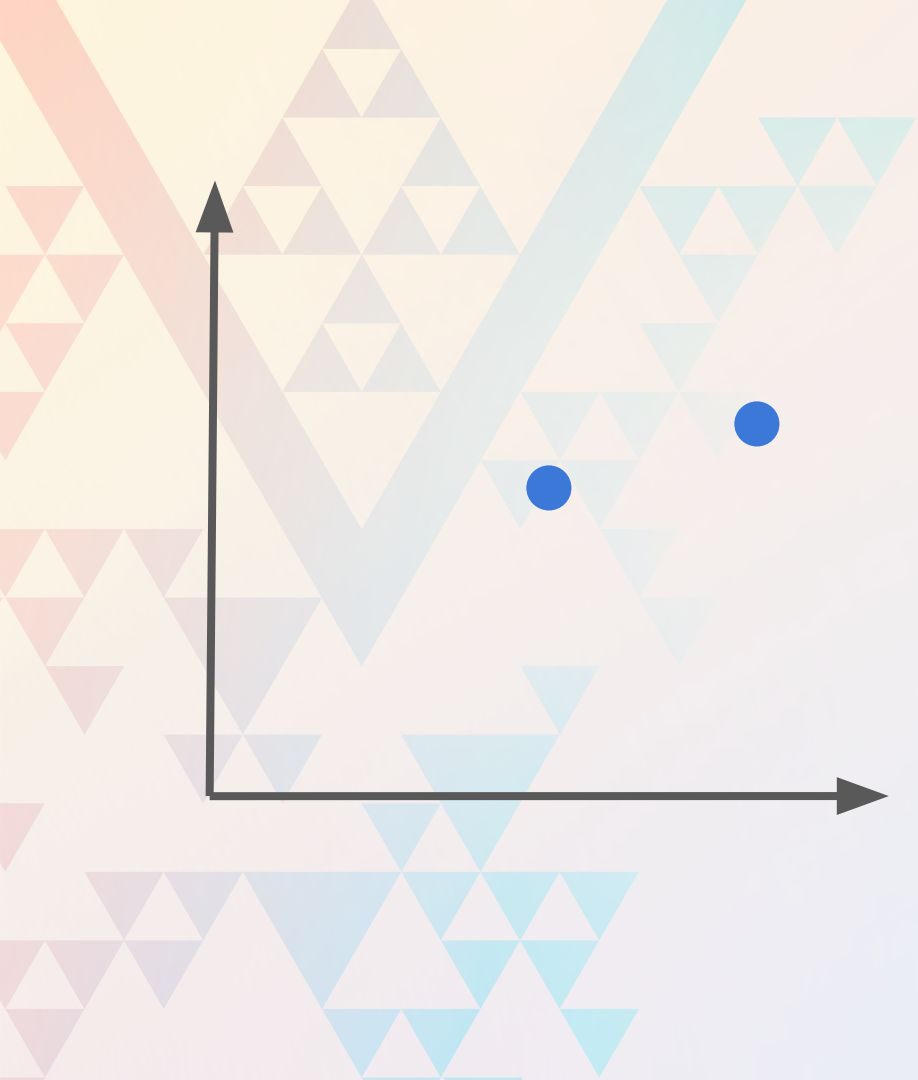


First, we need to understand how to share a secret key

Imagine you drew a line, and the y-intercept (star) represents your private key.

Now you draw 3 points on the line... and erase the line!

With only one point, it's impossible to redraw the line

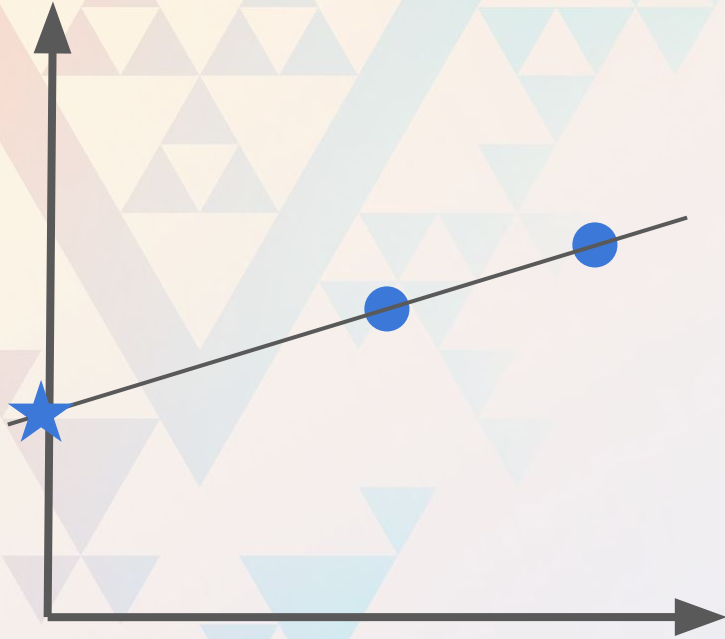


First, we need to understand how to share a secret key

Imagine you drew a line, and the y-intercept (star) represents your private key.

Now you draw 3 points on the line... and erase the line!

With only one point, it's impossible to redraw the line, but you can do so with any two points

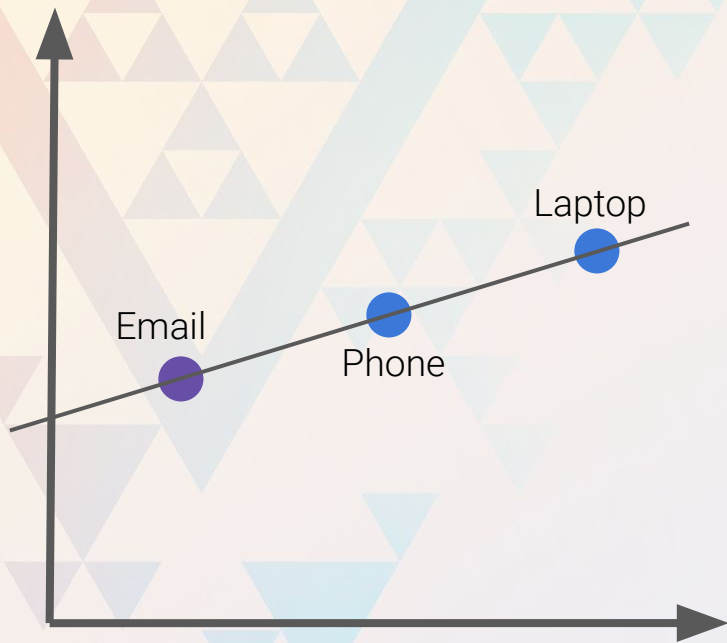


First, we need to understand how to share a secret key

Imagine you drew a line, and the y-intercept (star) represents your private key.

Now you draw 3 points on the line... and erase the line!

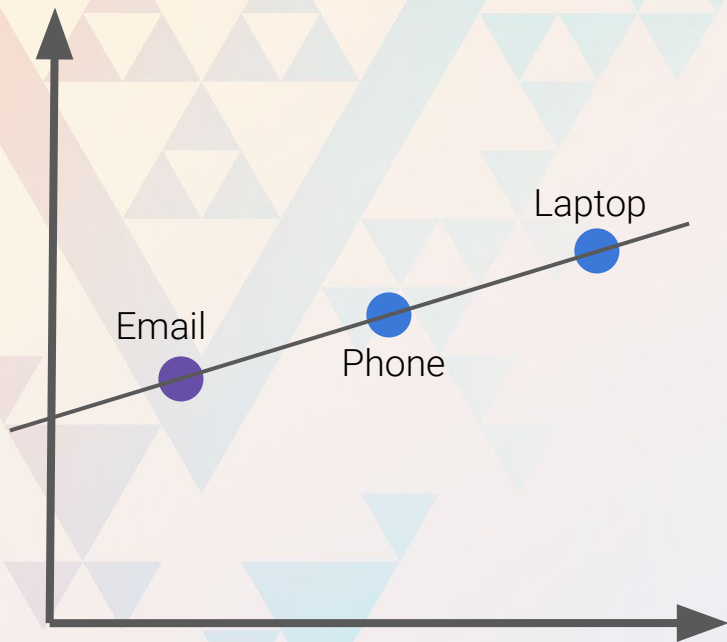
With only one point, it's impossible to redraw the line, but you can do so with any two points



First, we need to understand how to share a secret key

This is Shamir secret sharing, which splits a private key into multiple shares.

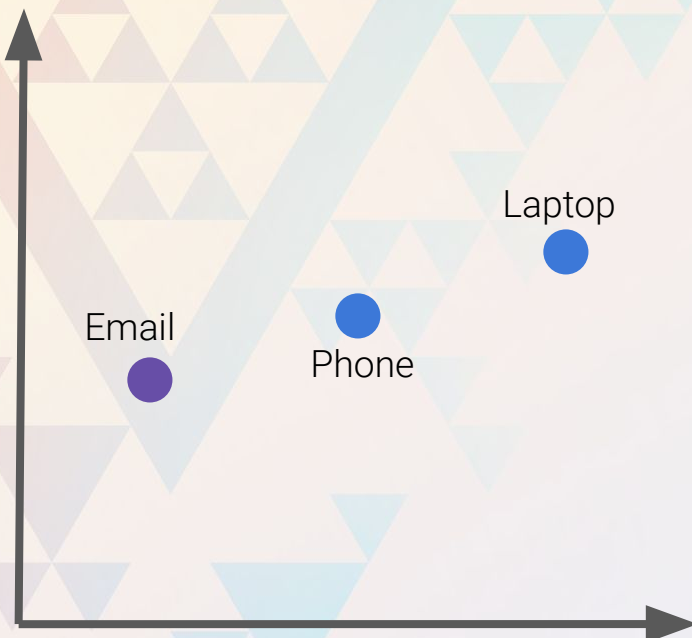
By tying each share to some user factor, we can get multi-factor authentication.



First, we need to understand how to share a secret key

This is Shamir secret sharing, which splits a private key into multiple shares.

By tying each share to some user factor, we can get multi-factor authentication.



So what is MPC?

MPC allows you generate and use the points (**shares**) to sign transactions without drawing the line (**private key**), ever!

No single point of failure.

For today we'll limit the scope to personal key management.



The binary dilemma

Non-custodial vs custodial

Non-custodial vs custodial

- We say a key is non-custodial if a user has full control of the majority of the shares
- And a key is custodial if 3rd parties (custodians) have control over a majority of the shares

It's pretty obvious that the above are exclusive properties.
You can't have two majorities.

(Special case: 2-out-of-2. User owns a share and 3rd party owns a share. 3rd party cannot sign without user consent but can censor transactions. Custodial? Non-custodial?)

Non-custodial vs custodial

Wouldn't it be nice if we had something better?

Non-custodial vs custodial

Wouldn't it be nice if we had something better?

Ideally, I want my private key to be non-custodial forever... until I've accidentally lost too many factors/died.. then it somehow becomes custodial... so at least my cryptoassets can be retrieved...

Non-custodial vs custodial

Wouldn't it be nice if we had something better?

Ideally, I want my private key to be non-custodial forever... until I've accidentally lost too many factors/died.. then it somehow becomes custodial... so at least my cryptoassets can be retrieved...

Sounds impossible?



Solving the impossible

Solution 1: Time-locked secret shares

Q: What is a time-lock?

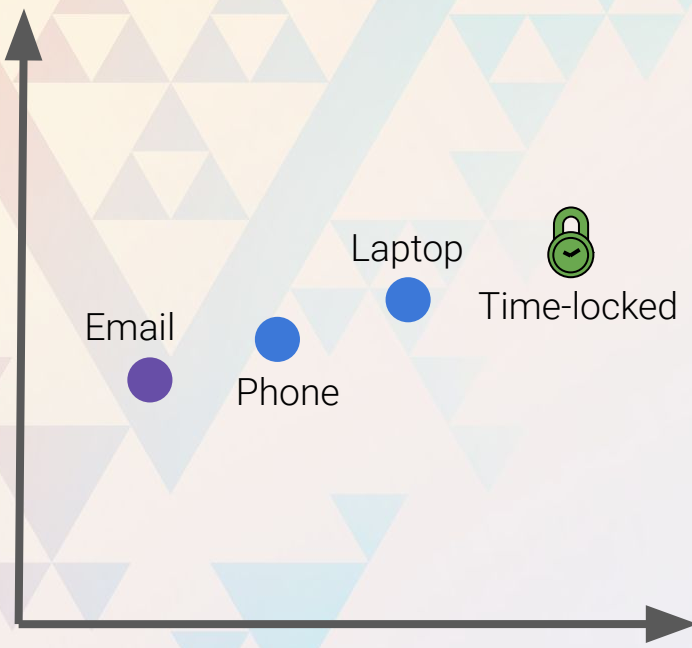
A: something that takes a long time to decrypt

A time-lock is something that should take a long amount of time to decrypt, and should not be able to be solved faster even with parallel computers.

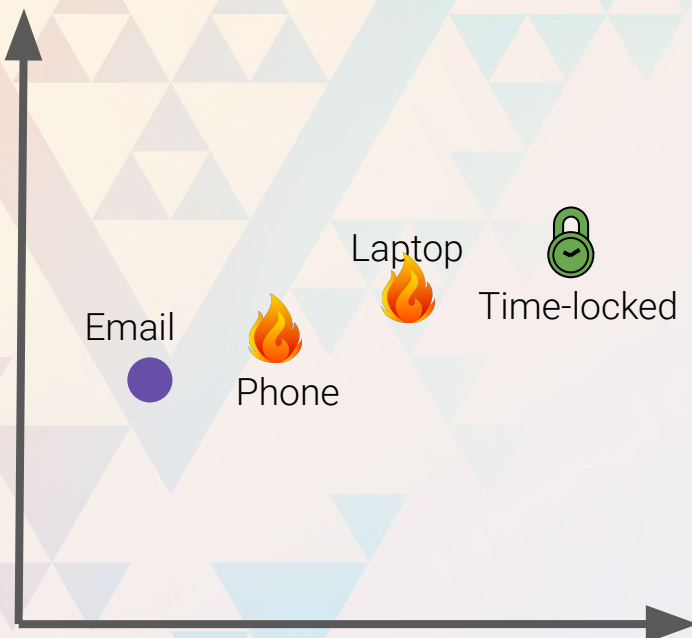
There are many variants: verifiable delay functions, verifiable delay encryption, homomorphic time-lock puzzles.

Most of these have properties that we don't need, and even properties we don't want. So concretely we'll be using the most basic one, which is repeated squaring.

It really doesn't matter. Just think: locked for a long time.

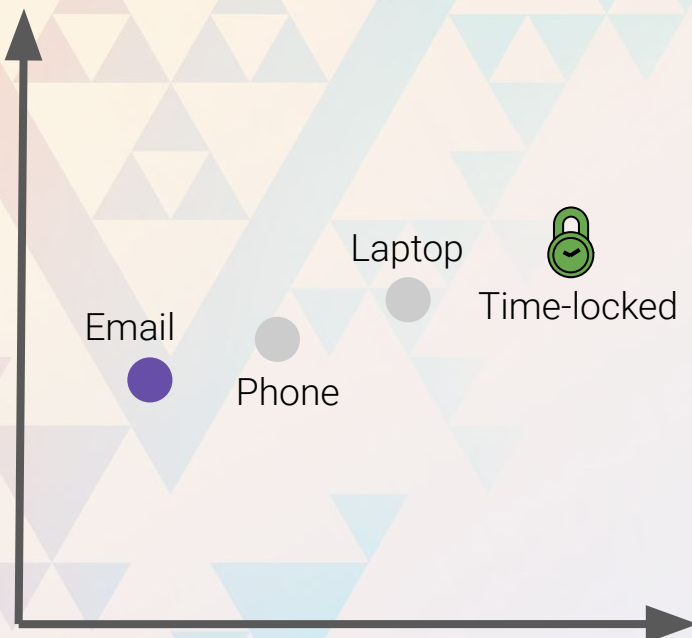


Let's just have an extra share that's time-locked



Let's just have an extra share that's time-locked

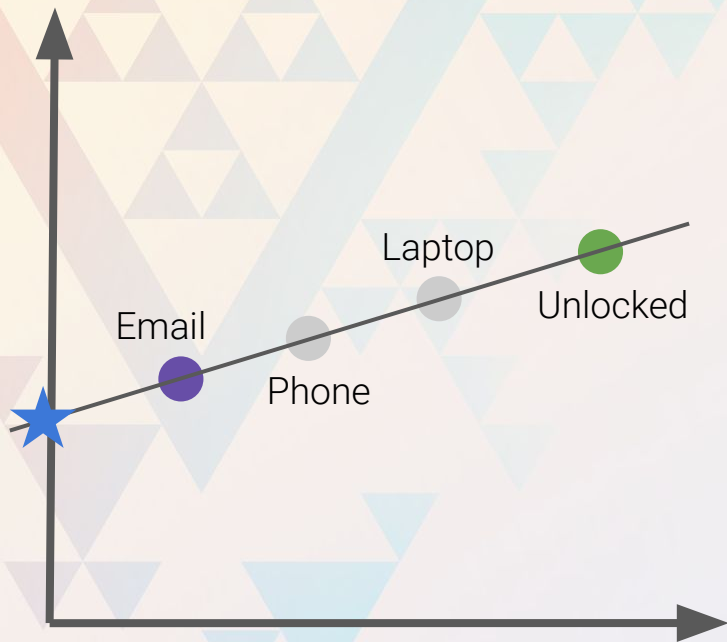
Let's say your house caught fire and you've lost both your laptop and phone.



Let's just have an extra share that's time-locked

Let's say your house caught fire and you've lost both your laptop and phone.

Luckily, you've prepared ahead of time and generated a fourth share that's time-locked for 1 month, before your house caught fire. You've put the time-locked share somewhere public (not in your house).



Let's just have an extra share that's time-locked

Let's say your house caught fire and you've lost both your laptop and phone.

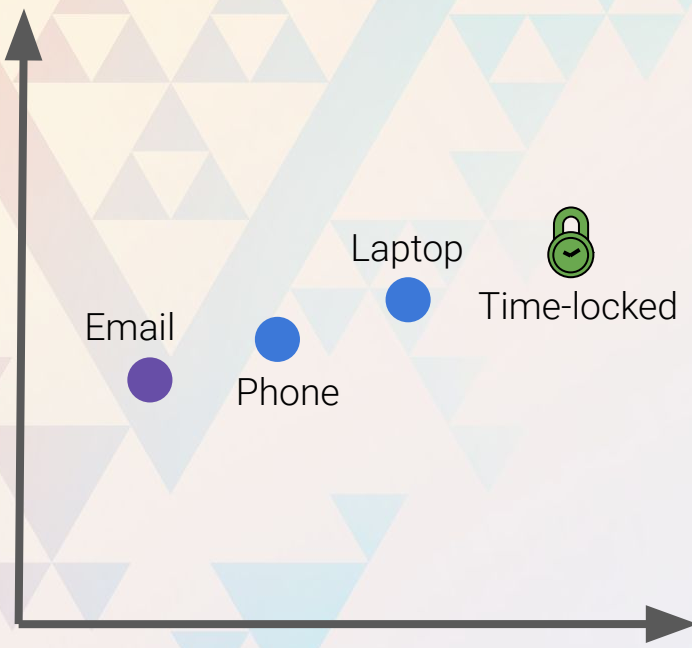
Luckily, you've prepared ahead of time and generated a fourth share that's time-locked for 1 month, before your house caught fire. You've put the time-locked share somewhere in the public (not in your house).

1 month from now the share unlocks and you can reconstruct your key since you still have access to your email. Perfect!

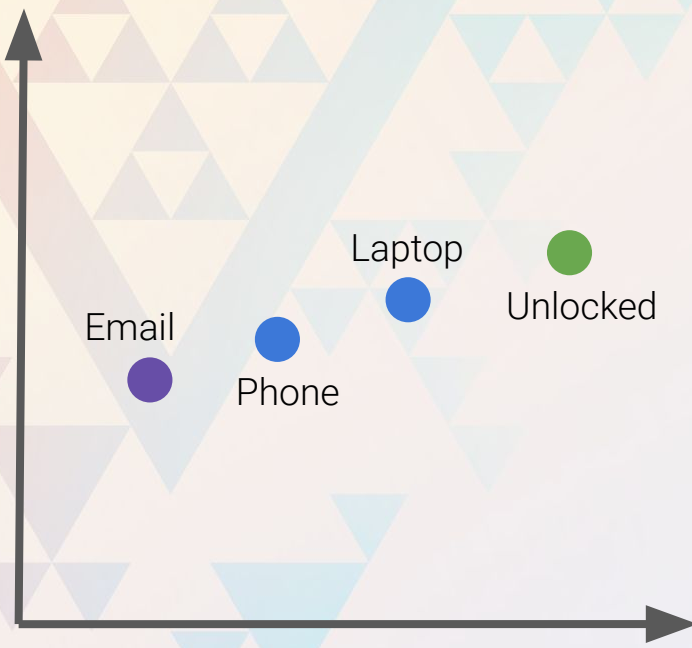


That won't work

Problem 1: Lowered security threshold

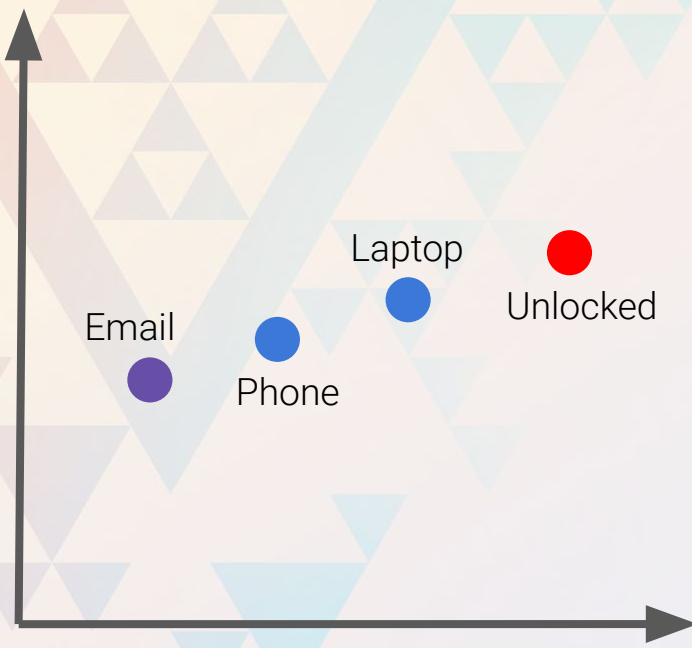


Now you're just 1 stolen factor away from losing your key



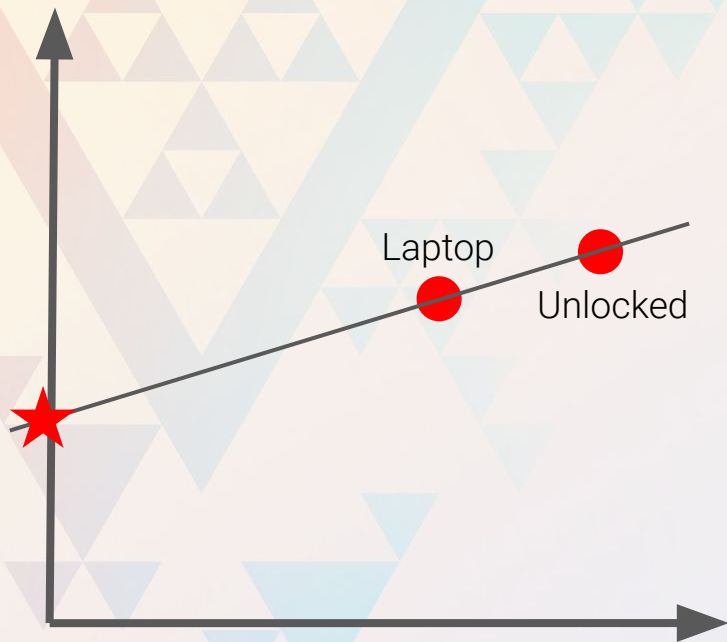
Now you're just 1 stolen factor away from losing your key

Let's say there's a patient adversary who waits a month for your time-locked share to unlock.



Now you're just 1 stolen factor away from losing your key

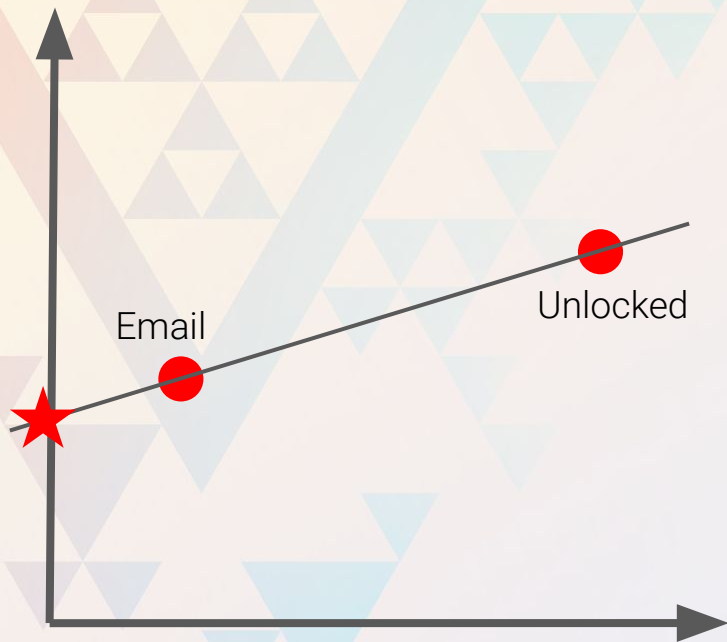
Let's say there's a patient adversary who waits a month for your time-locked share to unlock. Recall that you put this in a public location so it doesn't burn down with your house. So the adversary can access it easily.



Now you're just 1 stolen factor away from losing your key

Let's say there's a patient adversary who waits a month for your time-locked share to unlock. Recall that you put this in a public location so it doesn't burn down with your house. So the adversary can access it easily.

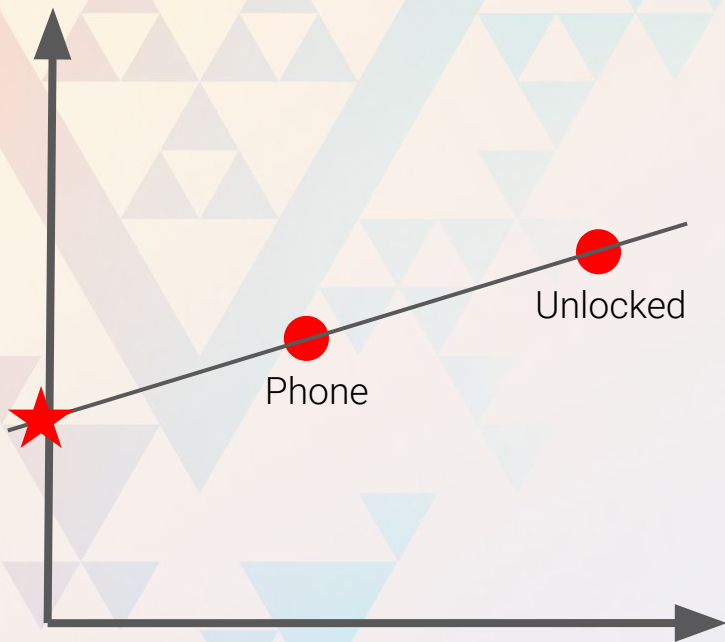
Literally any other factor being compromised leads to key loss



Now you're just 1 stolen factor away from losing your key

Let's say there's a patient adversary who waits a month for your time-locked share to unlock. Recall that you put this in a public location so it doesn't burn down with your house. So the adversary can access it easily.

Literally any other factor being compromised leads to key loss



Now you're just 1 stolen factor away from losing your key

Let's say there's a patient adversary who waits a month for your time-locked share to unlock. Recall that you put this in a public location so it doesn't burn down with your house. So the adversary can access it easily.

Literally any other factor being compromised leads to key loss

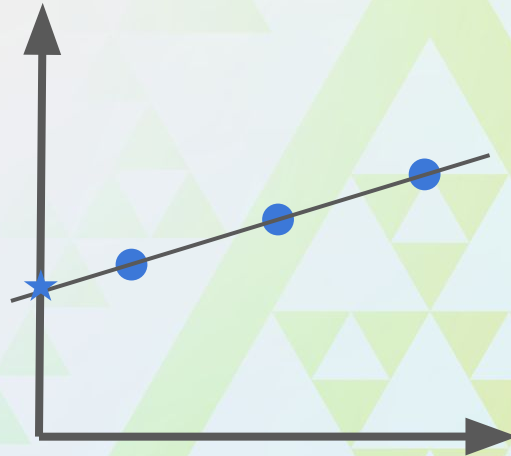


Rotate your secret shares

Solution 2: Time-locked secret shares with key refresh

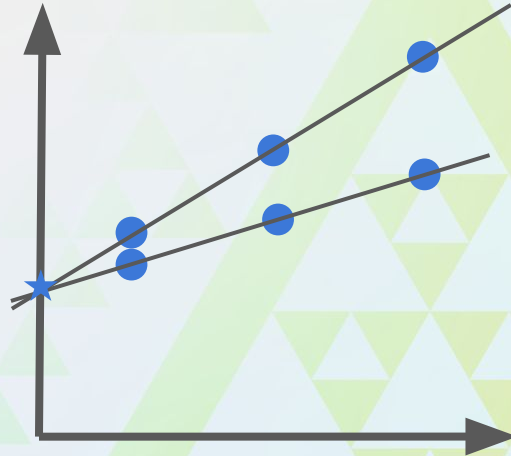
Q: What is key refresh?

A: Change all the shares
but keep the private key
the same



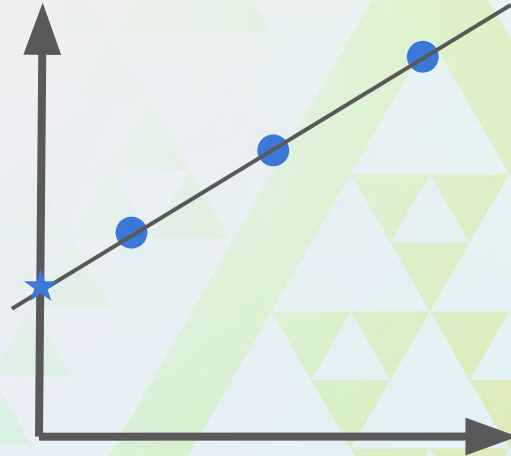
Q: What is key refresh?

A: Change all the shares
but keep the private key
the same



Q: What is key refresh?

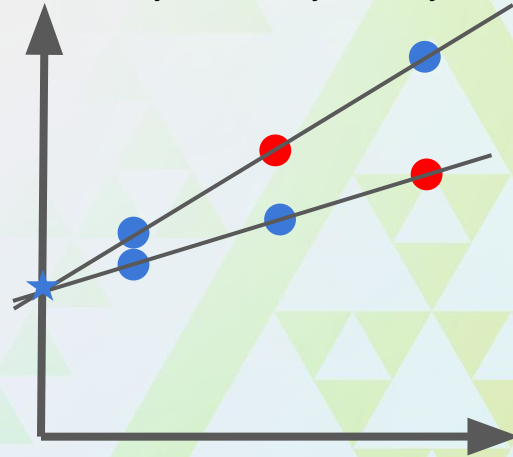
A: Change all the shares
but keep the private key
the same



Q: What is key refresh?

A: Change all the shares but keep the private key the same

As long as two factors aren't compromised before a key refresh, your key is safe



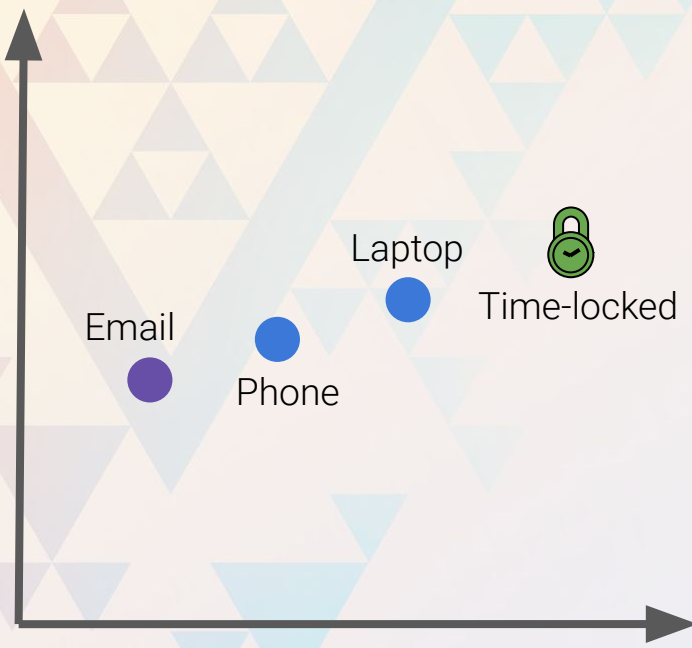
Q: What is

**DISCLAIMER:
THIS IS FOR ILLUSTRATION
PURPOSES ONLY, PLEASE
DON'T DO IT EXACTLY LIKE
THIS IN PRODUCTION**

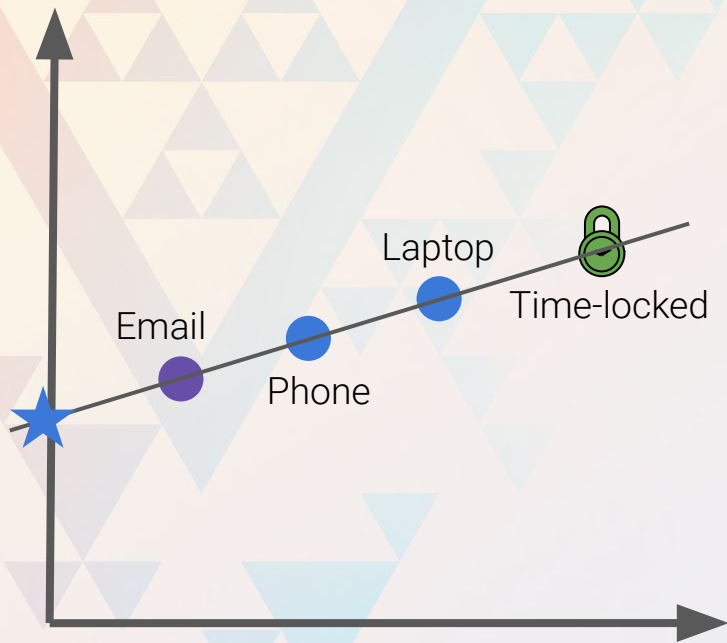
A: Change all the shares
private key

isn't compromised
key is safe



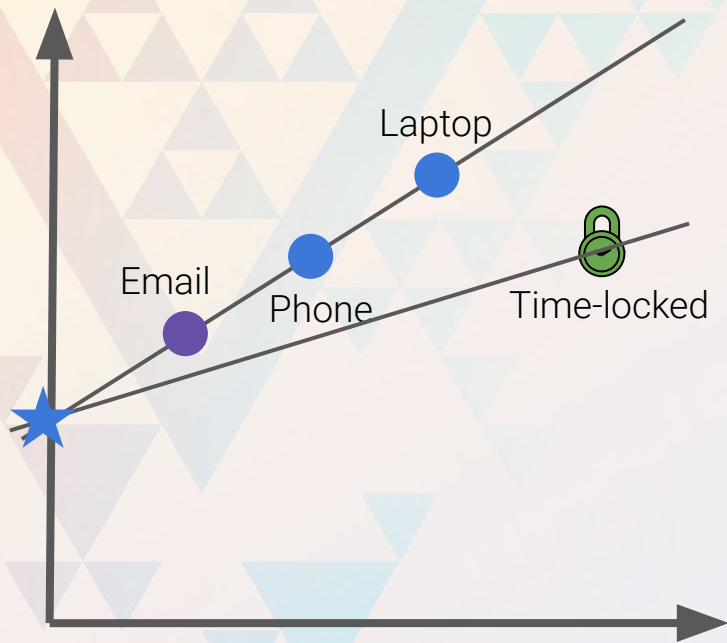


What if we key refresh
before the time-lock is
up?



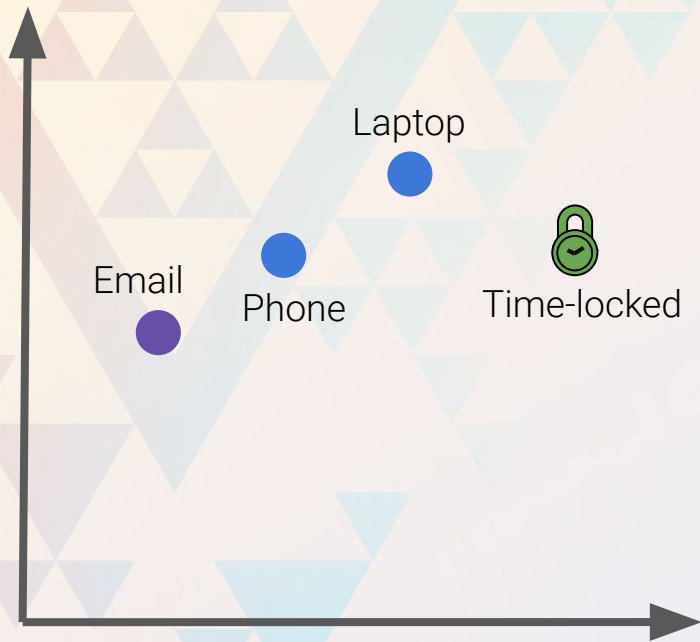
What if we key refresh
before the time-lock is
up?

We do a key refresh whenever we use our wallet / once a week, whichever is longer.



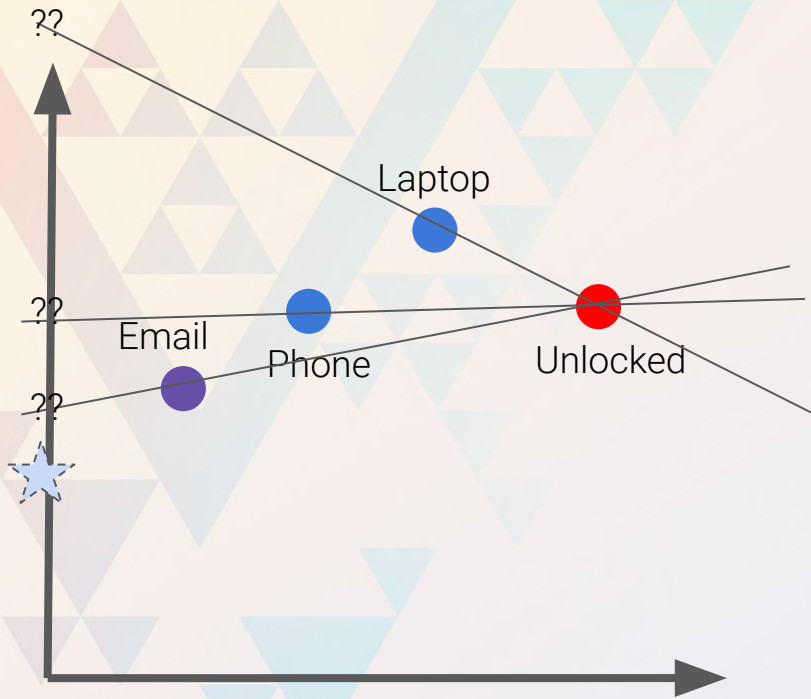
What if we key refresh
before the time-lock is
up?

We do a key refresh whenever we use our wallet / once a week, whichever is longer.



What if we key refresh
before the time-lock is
up?

We do a key refresh whenever we use our wallet / once a week, whichever is longer.



What if we key refresh before the time-lock is up?

We do a key refresh whenever we use our wallet / once a week, whichever is longer.

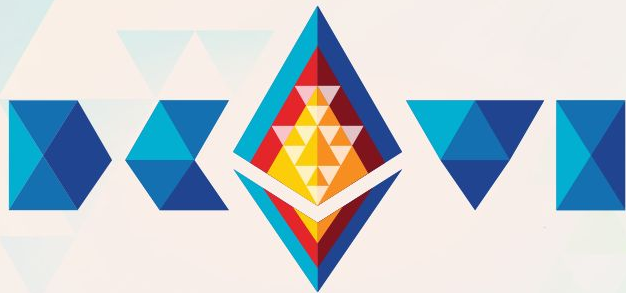
When the time-lock unlocks, the adversary compromises it but even if they steal other factors, the share wasn't generated on the same line, so it's not compatible with the other shares.

This seems to work!

This setup seems to have a lot of the properties that we want.

- Normal operation: attacker can solve time-lock but can't use it with other shares since key refresh has occurred
 - Interesting property where time-lock puzzles being expensive to evaluate is a feature, expensive to attack with low probability of success deters attackers. (homomorphic time-lock puzzles are NOT useful here)
- Catastrophic loss: time-locked share is revealed, share refresh doesn't happen because user has no access to his other shares.
 - custodial share + time-locked share reconstructs key.
- Very similar to a dead man switch in smart contract wallets.

We've created a non-custodial setup that degrades into a custodial setup on loss!



We're done!
Thank you for coming to my TED talk!

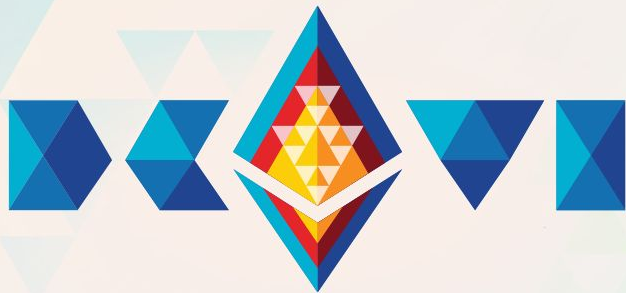
Leonard Tan

CTO, Web3Auth

leonard@web3auth.io



@bluzuli



Just kidding.

Leonard Tan

CTO, Web3Auth

leonard@web3auth.io

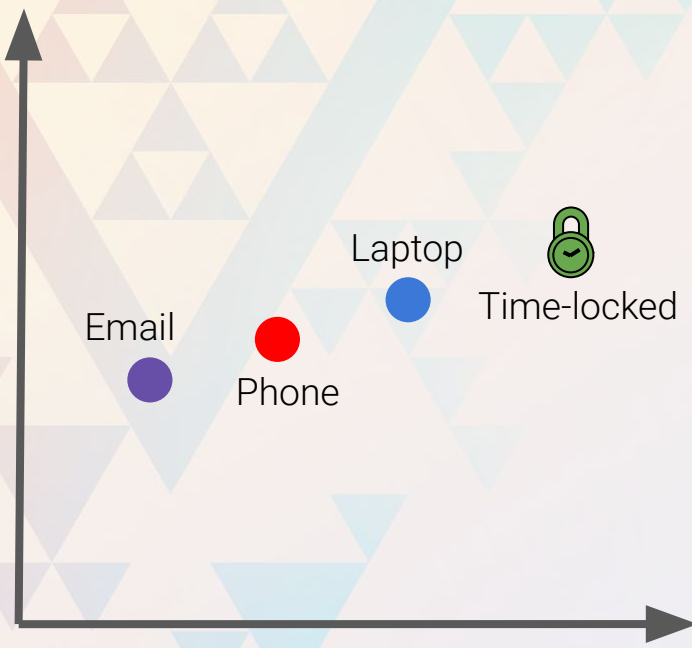


@bluzuli

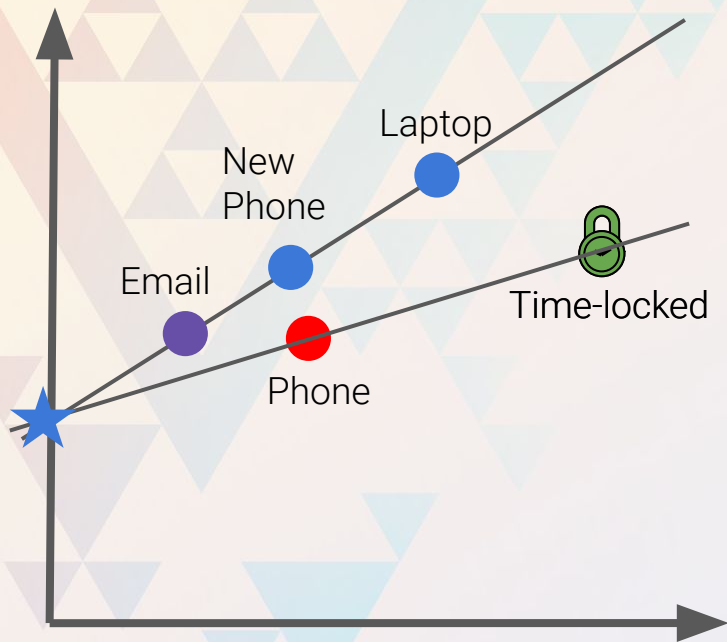


Just kidding, this is full of holes.

Problem 2: Phone share is stolen, then
time-locked share unlocks

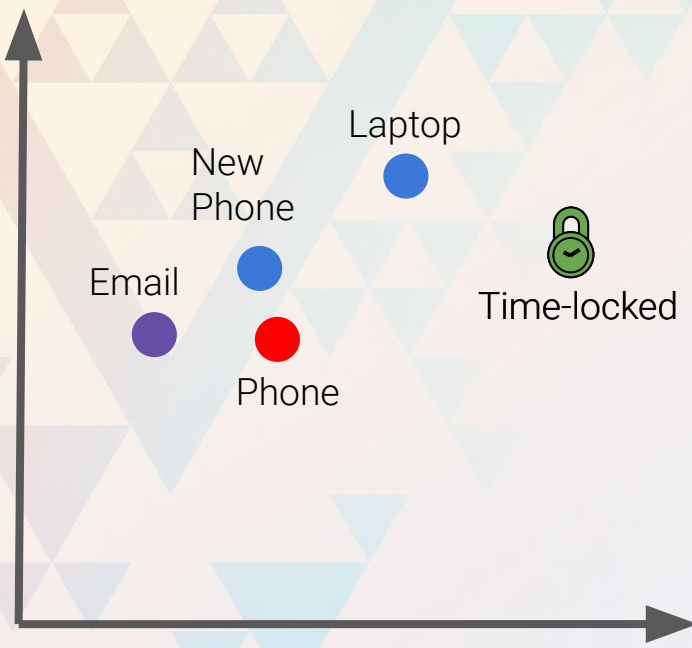


What happens if your phone gets stolen?



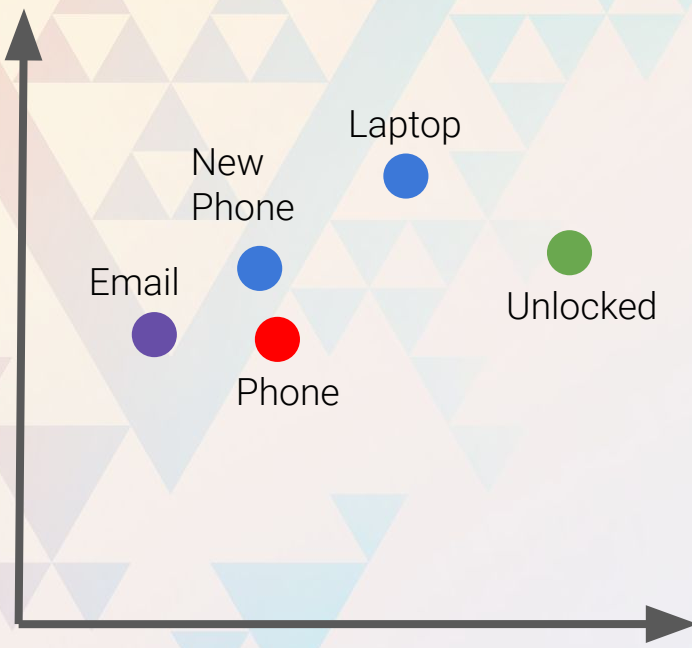
What happens if your phone gets stolen?

Let's try key refresh.



What happens if your phone gets stolen?

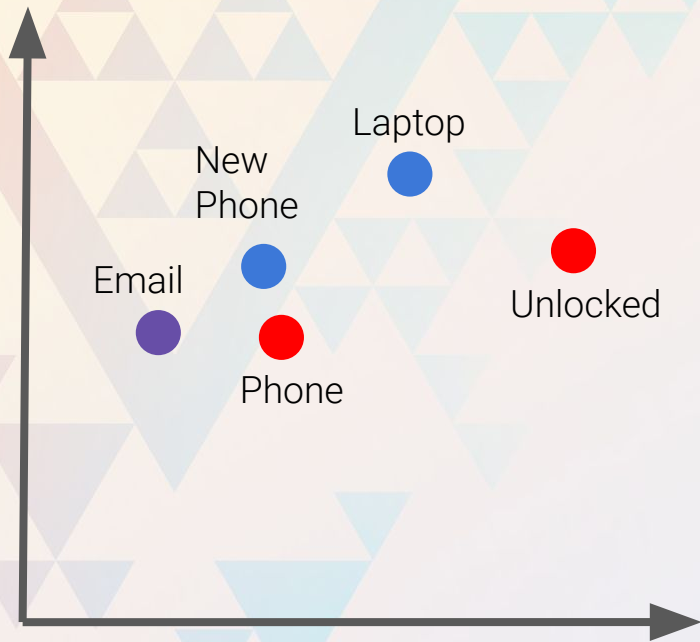
Let's try key refresh.



What happens if your phone gets stolen?

Let's try key refresh.

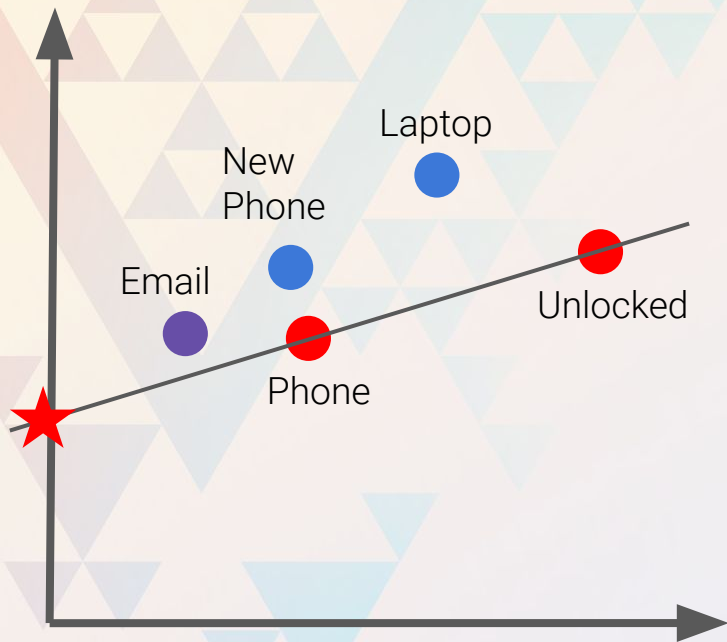
A month passes...



What happens if your phone gets stolen?

Let's try key refresh.

A month passes... and our time-locked share becomes public, and an adversary gets their hands on it.



What happens if your phone gets stolen?

Let's try key refresh.

A month passes... and our time-locked share becomes public, and an adversary gets their hands on it.

Key gone.



Stop making more shares.

Solution 3: The time-locked share IS my
phone share. Now I'm safe.



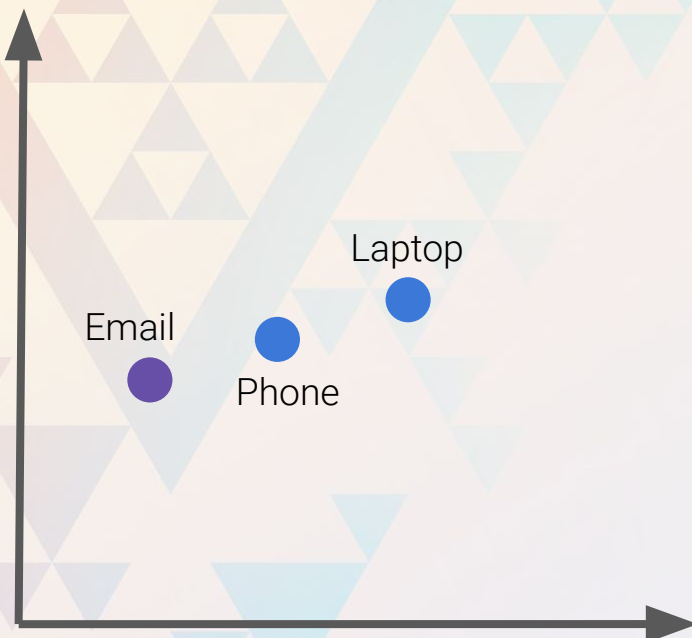
Dammit.

Problem 4: Laptop share is stolen, then
time-locked share unlocks

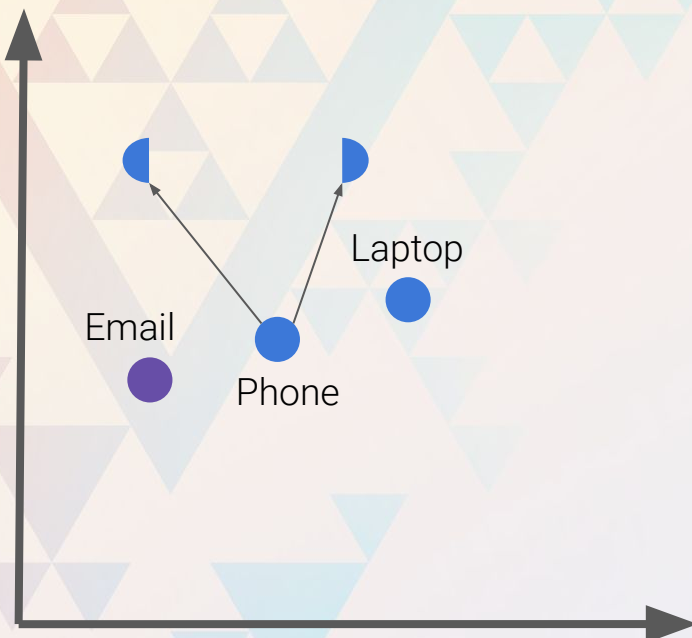


Make subshares instead

Solution 4: Split device share in two:
a time-locked and a custodial subshare

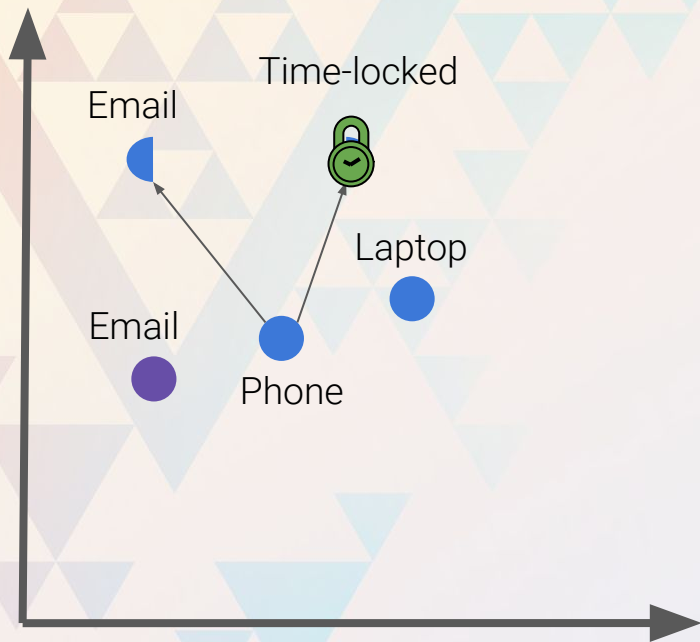


Split a device share into two subshares



Split a device share into two subshares

After doing that...



Split a device share into two subshares

After doing that...

Generate a time-lock on one half, and encrypt the second half under the share governed by the email.

Let's analyze this set up

during loss:

= email share + (email subshare + time-locked subshare)

= email share + phone share

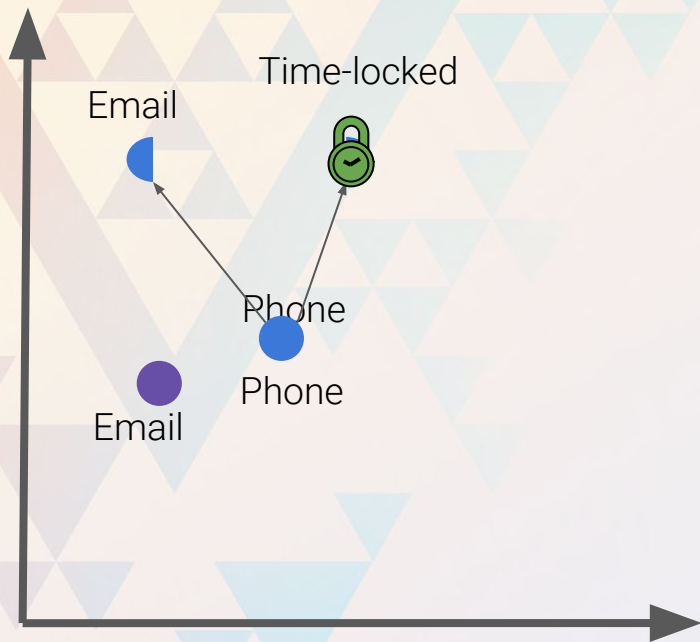
= key → recovery still works!

If adversary steals laptop or phone, the adversary still needs to compromise the email subshare even if the time-locked subshare is revealed. As long as the user does key refresh and the email share is refreshed, the adversary cannot get the key.



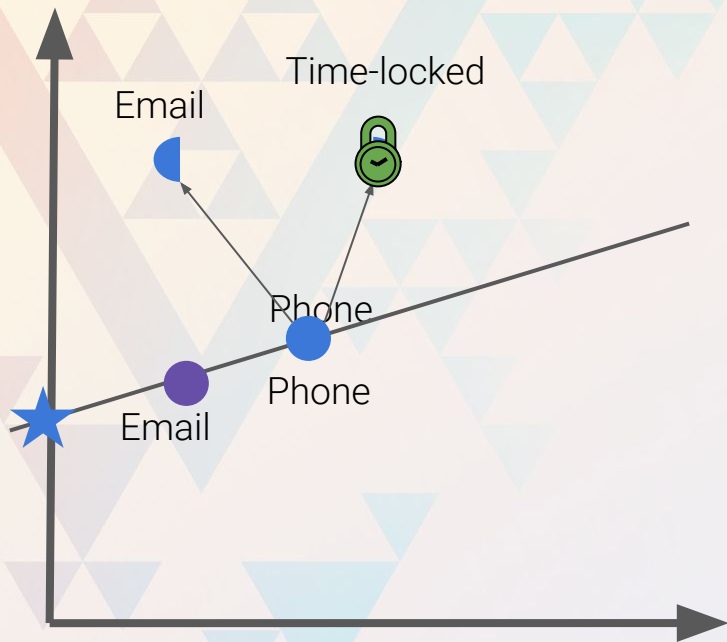
The non-deletion problem.

Problem 5: Email provider doesn't delete shares during key-refresh



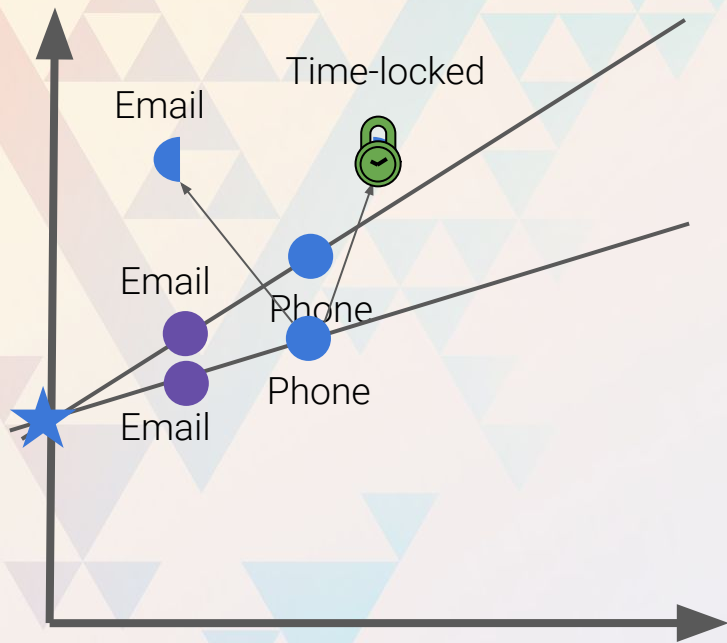
What happens if the custodial factor doesn't delete shares?

Let's go back to the scenario of a key refresh, but the email provider (custodial!) doesn't delete their previous share



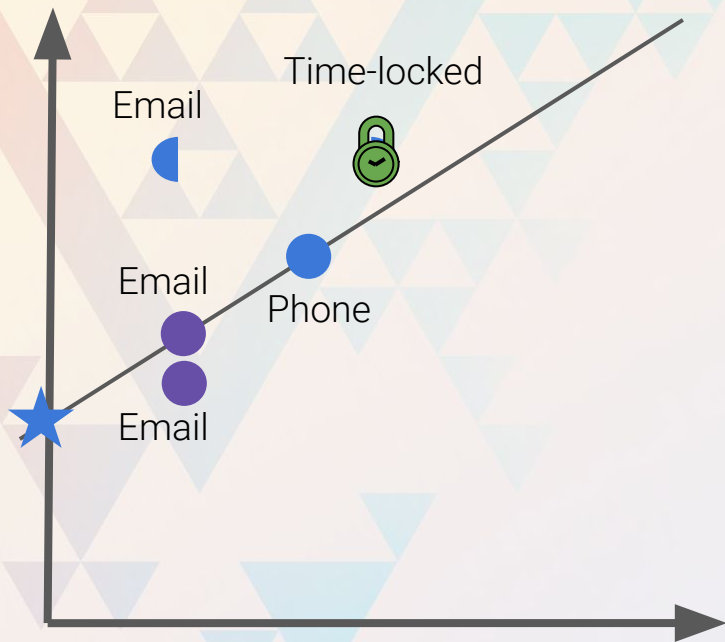
What happens if the custodial factor doesn't delete shares?

Let's go back to the scenario of a key refresh, but the email provider (custodial!) doesn't delete their previous share



What happens if the custodial factor doesn't delete shares?

Let's go back to the scenario of a key refresh, but the email provider (custodial!) doesn't delete their previous share

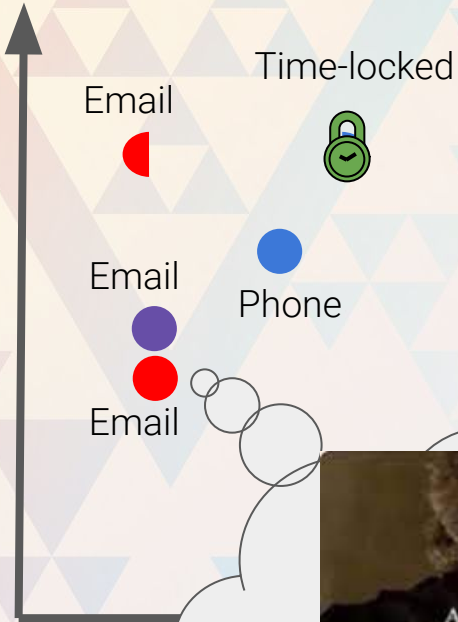


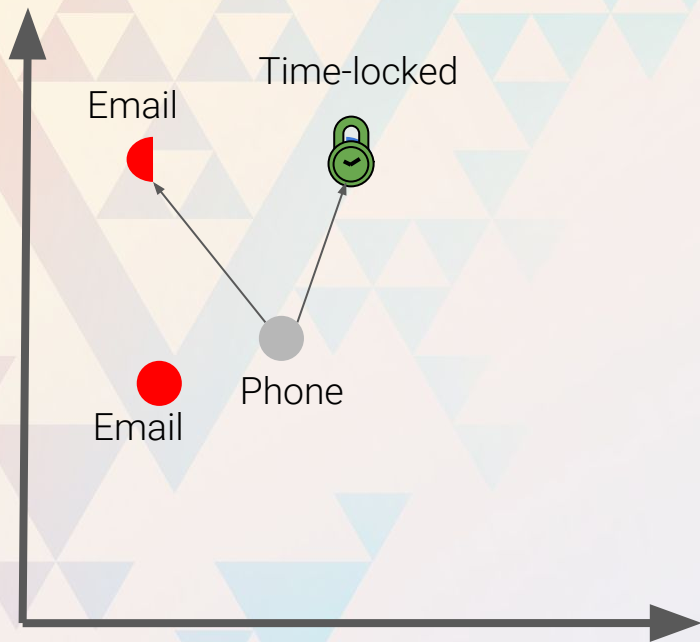
What happens if the custodial factor doesn't delete shares?

Let's go back to the scenario of a key refresh, but the email provider (custodial!) doesn't delete their previous share

What happens if the custodial factor doesn't delete shares?

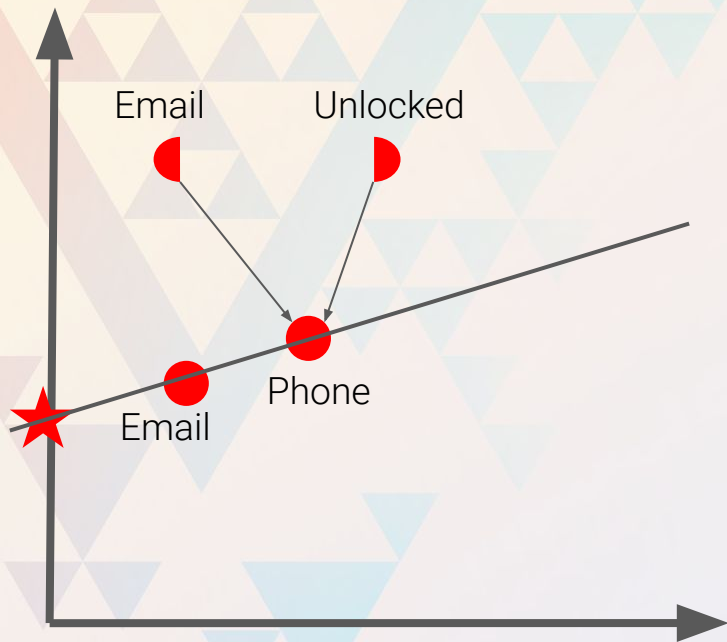
Let's go back to the scenario of a key refresh, but the email provider (custodial!) doesn't delete their previous share





What happens if the custodial factor doesn't delete shares?

Let's go back to the scenario of a key refresh, but the email provider (custodial!) doesn't delete their previous share



What happens if the custodial factor doesn't delete shares?

Let's go back to the scenario of a key refresh, but the email provider (custodial!) doesn't delete their previous share

After a month passes, they have enough key material to reconstruct the key.



Custodians bad! No custodial factors!

Solution 6: Let's just get rid of all
custodians / 3rd parties!



We're trying to solve for the disaster case where you lose ALL your user-owned factors? Do you really want all the factors to be lost?

Problem 6: It's not possible to get recovery from catastrophic loss without 3rd-parties having threshold-1 shares.

Custodians need to control threshold - 1 shares, if we want recoverability from catastrophic loss

Fundamentally, if custodians only control threshold-2 shares, with the time-locked share unlocking, they would only have threshold-1 shares, which does not meet the threshold. So a user-owned share has to be used, but that's impossible since in this scenario, the user has lost all their shares / died (and so they cannot access all their shares).

Custodians need to control threshold - 1 shares, if we want recoverability from catastrophic loss

Fundamentally, if custodians only control threshold-2 shares, with the time-locked share unlocking, they would only have threshold-1 shares, which does not meet the threshold. So a user-owned share has to be used, but that's impossible since in this scenario, the user has lost all their shares / died (and so they cannot access all their shares).

So we haven't really solved the non-deletion problem. We need 3rd-party custodians to have threshold-1 shares, but we can't guarantee deletion. So in 1 month, assuming the custodian is malicious, they will unlock the time-locked share and get the private key...



Final solution. For now.

Solution 7: 1-out-of-n deletion.

The more green, the better. Let us explore the categories in more detail:

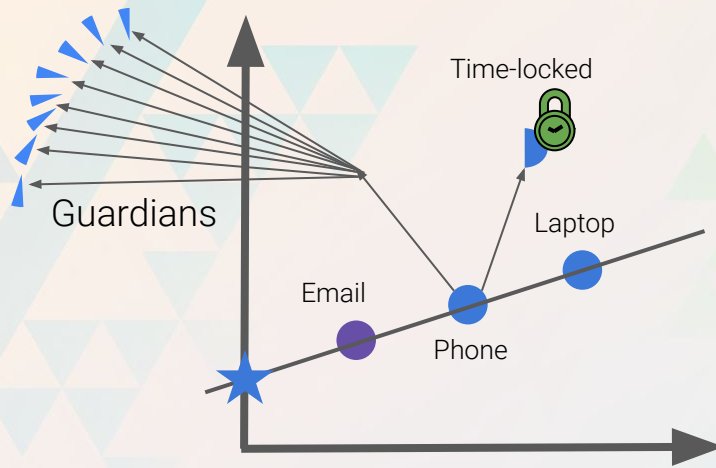
- **1 of 1:** there is exactly one actor, and the system works if (and only if) that one actor does what you expect them to. This is the traditional "centralized" model, and it is what we are trying to do better than.
- **N of N:** the "dystopian" world. You rely on a whole bunch of actors, *all* of whom need to act as expected for everything to work, with no backups if any of them fail.
- **N/2 of N:** this is how blockchains work - they work if the majority of the miners (or PoS validators) are honest. Notice that N/2 of N becomes significantly more valuable the larger the N gets; a blockchain with a few miners/validators dominating the network is much less interesting than a blockchain with its miners/validators widely distributed. That said, we want to improve on even this level of security, hence the concern around surviving 51% attacks.
- **1 of N:** there are many actors, and the system works as long as at least one of them does what you expect them to. Any system based on fraud proofs falls into this category, as do trusted setups though in that case the N is often smaller. Note that you do want the N to be as large as possible!
- **Few of N:** there are many actors, and the system works as long as at least some small fixed number of them do what you expect them do. *Data availability checks* fall into this category.
- **0 of N:** the systems works as expected without any dependence whatsoever on external actors. Validating a block by checking it yourself falls into this category.

Taken from vitalik.ca

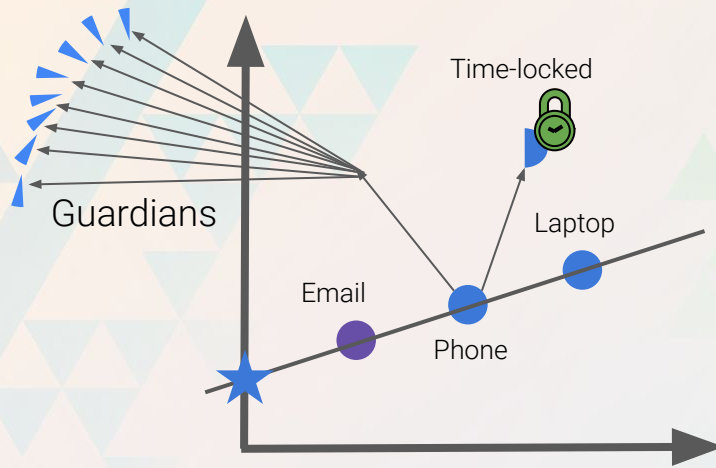
Trust that there is at
least 1 party that
follows the rules

1-out-of-n deletion

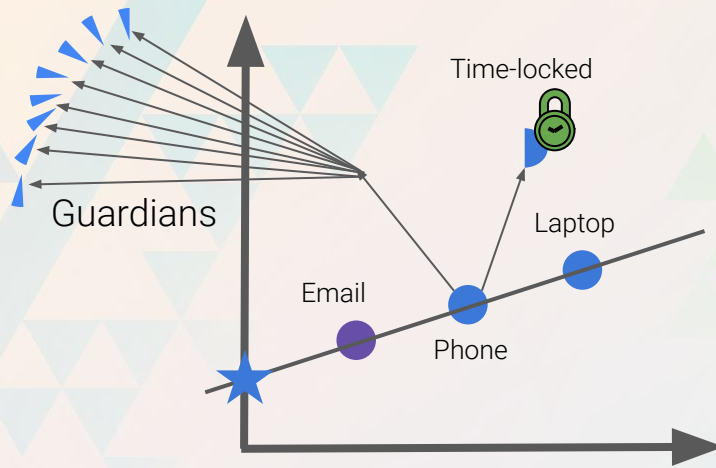
Have the custodial subshare be further split across 100 independent custodians/guardians and trust that one will delete it during refresh. This isn't ideal, but it's close enough, and if the set gets bigger (~1000) it's pretty close to trustless. Doing n-out-of-n secret sharing is very fast, so it's also practical.



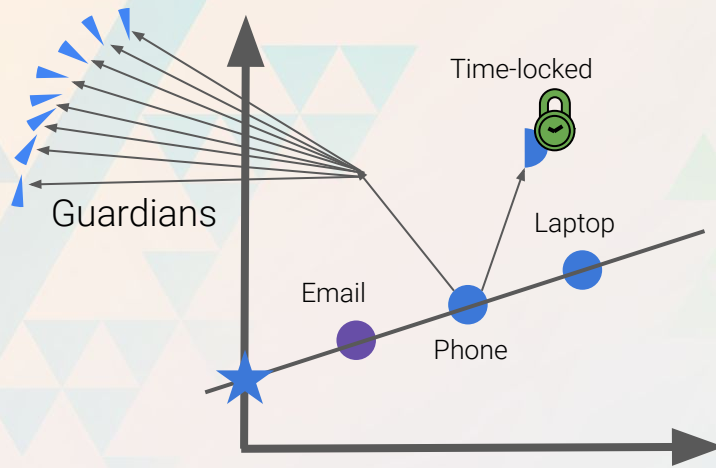
During normal operation, key refresh + 1-out-of-n deletion defends against external adversary who accesses one factor.



During normal operation, email provider only has access to one factor and one of the subshares (time-locked), since the other one is deleted



During catastrophic loss, when all user factors are gone, the email factor + time-locked subshare + guardians subshare is sufficient for key recovery.



There. A bit more complex, but much better than what we started with.

<https://gist.github.com/tetratorus/303f35ee0871f40e2b91a085d03ca97e>




Or just use a smart contract wallet
with a dead man switch.



Or just use a smart contract wallet
with a dead man switch.

Not everything needs MPC y'know.

The background features a complex geometric pattern of overlapping triangles and lines in various colors including orange, yellow, green, and blue. The triangles are arranged in a way that creates a sense of depth and movement, with some appearing as solid shapes and others as outlines or semi-transparent layers. The overall effect is a vibrant, abstract design.

Open problems:
guardian grieving attacks?
retrieval from guardians?



Thank you!

Leonard Tan

CTO, Web3Auth

leonard@web3auth.io



@bluzuli