Minimal Design for Decentralized Wallet

Omer
Shlomovits

Motivation

* Imagine we had a private key management system where:

- *No single point of failure*
- *Move of assets (signing) cannot happen without Owner approval*
- *Recovery is possible at all times*

Our Heroes

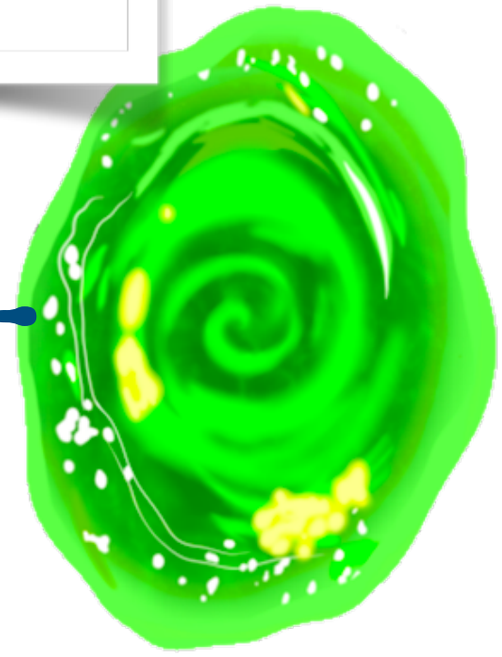
Rick

Morty

The Journey

Mixed Model

Verifiable Social Recovery



KMS

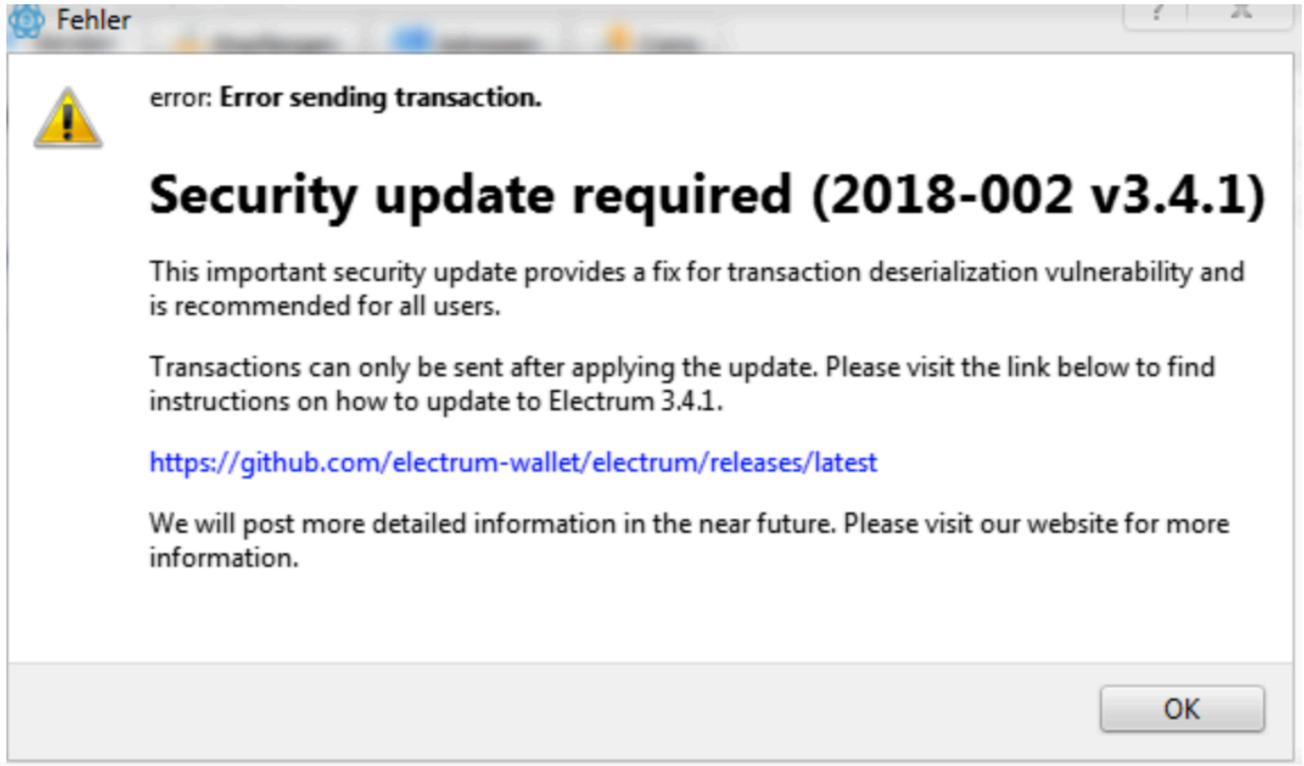
TMOVE

Key Management System

* Three main functionalities in the context of blockchains:
  - Key generation and custody
  - Signatures
  - Key backup and recovery

Problem: Key Management is hard

* Side channel attacks
* social engineering attacks
* human errors
* etc…



Fake alert created by the attacker (via Electrum GitHub page)

WALLET.FAIL

Poof goes your crypto …

devops199 commented 22 hours ago • edited

I accidentally killed it.

https://etherscan.io/address/0x863df6bfa4469f3ead0be8f9f2aae51c91a907b4

KZ

Trusted Party to the rescue?

Trusted Party to the rescue?

2018: A Record-Breaking Year for **Crypto Exchange** Hacks
CoinDesk - 29 Dec 2018
From the number of **cryptocurrency exchange** hacks, to the amount of assets that were stolen, to the largest **exchange hack** of all-time, **crypto** ...

How Hackers Stole $1B From **Cryptocurrency Exchanges** In 2018
Forbes - 31 Dec 2018
The methodology behind the biggest **cryptocurrency hack** of the year has never been made public. However, the Japan Times reported at the ...

Trusted Party to the rescue?

2018: A Record-Breaking Year for **Crypto Exchange** Hacks
CoinDesk - 29 Dec 2018
From the number of **cryptocurrency exchange** hacks, to the amount of assets that were stolen, to the largest **exchange hack** of all-time, **crypto** ...

How Hackers Stole $1B From **Cryptocurrency Exchanges** In 2018
Forbes - 31 Dec 2018
The methodology behind the biggest **cryptocurrency hack** of the year has never been made public. However, the Japan Times reported at the ...

How can we optimize on keys security without compromising on keys usability ?

KZ

From single to distributed keys

- Enter threshold cryptography

From single to distributed keys

- Enter threshold cryptography
- Given n parties, we divide the key management responsibilities
  - Distributed key generation
  - Signing requires cooperation of t out of n

From single to distributed keys

- Enter threshold cryptography
- Given n parties, we divide the key management responsibilities
  - Distributed key generation
  - Signing requires cooperation of t out of n
- Efficient protocols exists
  - Threshold ECDSA [GG18, DKLS18, LNR18]

Sounds a lot like
Multisig to me

Multisig vs Threshold Signing

https://medium.com/kzen-networks/threshold-signatures-private-key-the-next-generation-f27b30793b

- Max number of parties
- Interactiveness
- Rotation
- Access policy privacy
- Chain support
- Low cost
- Efficiency (communication/ computation)

KZ

Multisig vs Threshold Signing

https://medium.com/kzen-networks/threshold-signatures-private-key-the-next-generation-f27b30793b

* Max number of parties
* Interactiveness
* Rotation
* Access policy privacy
* Chain support
* Low cost
* Efficiency (communication/ computation)

Mixed Model

Mixed Model

* Roles
  - *Owner x 1*
  - *Service Providers*

Mixed Model {t=n=2}

* Roles
  - *Owner x 1*
  - *Service Providers x 1*

Mixed Model {t=n=2}

* Roles
  - *Owner x 1*
  - *Service Providers x 1*

* System Requirements
  - ☑ *No single point of failure*
  - ☑ *Move of assets (signing) cannot happen without Owner approval*
  - *Recovery is possible at all times* ?

KZ

Choosing Parameters {t,n}

- Depends on the adversary model and specific use case different access structures can be considered

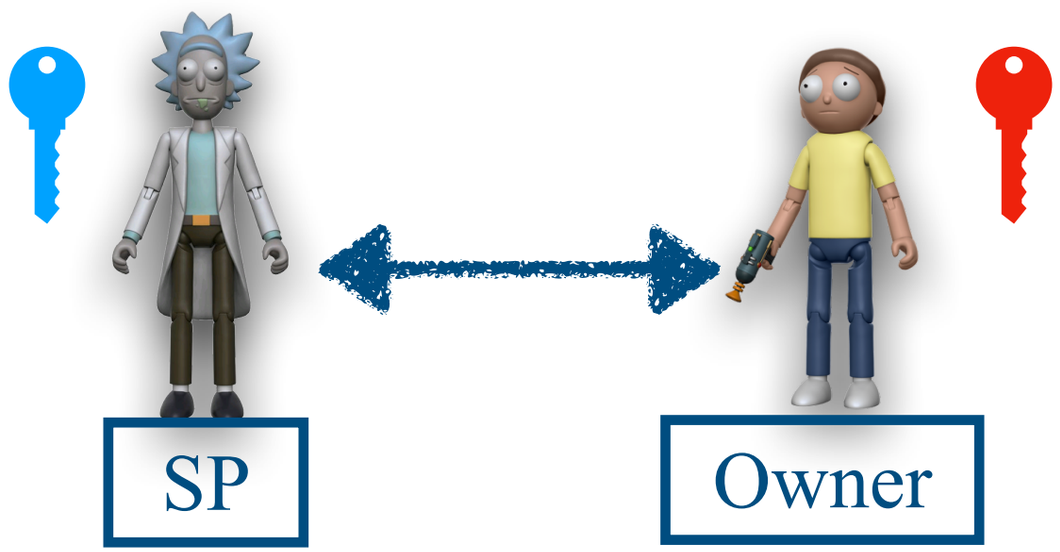
- Axiom: assuming SP is motivated solely by Economical Gain, We cannot avoid the Recovery problem

- Fact: two-party protocols are simpler than multi-party protocols

Recovery in Mixed Model{t=n=2}

Recovery in Mixed Model{t=n=2}

- Recovery in the two party setting can mean:
  - Self recovery: Owner's secret share
  - Counter party recovery: SP secret share

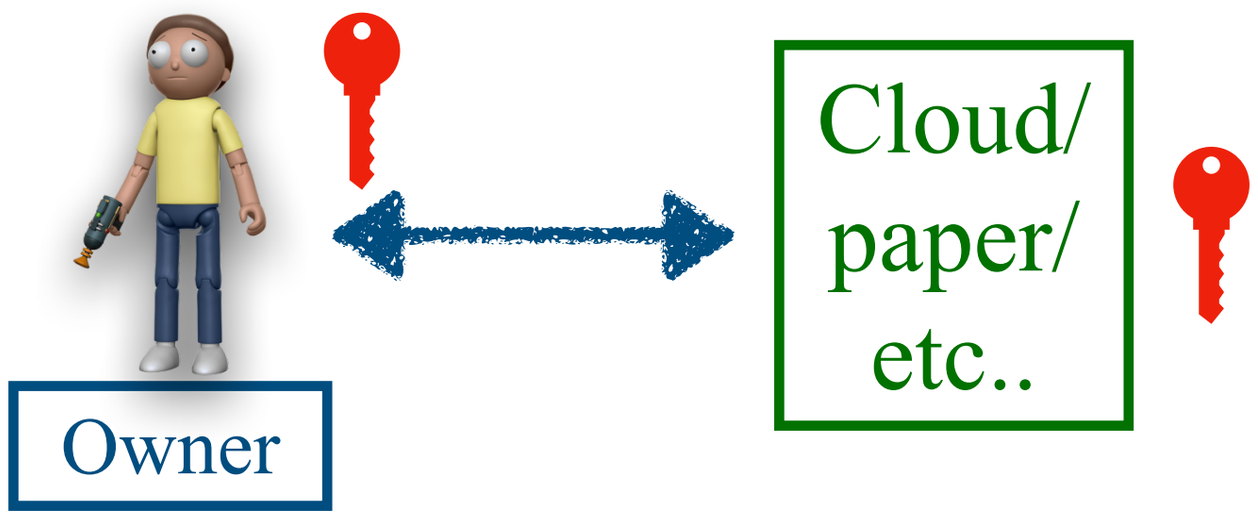


$$sk = f(\ \ ,\ \ )$$

KZ

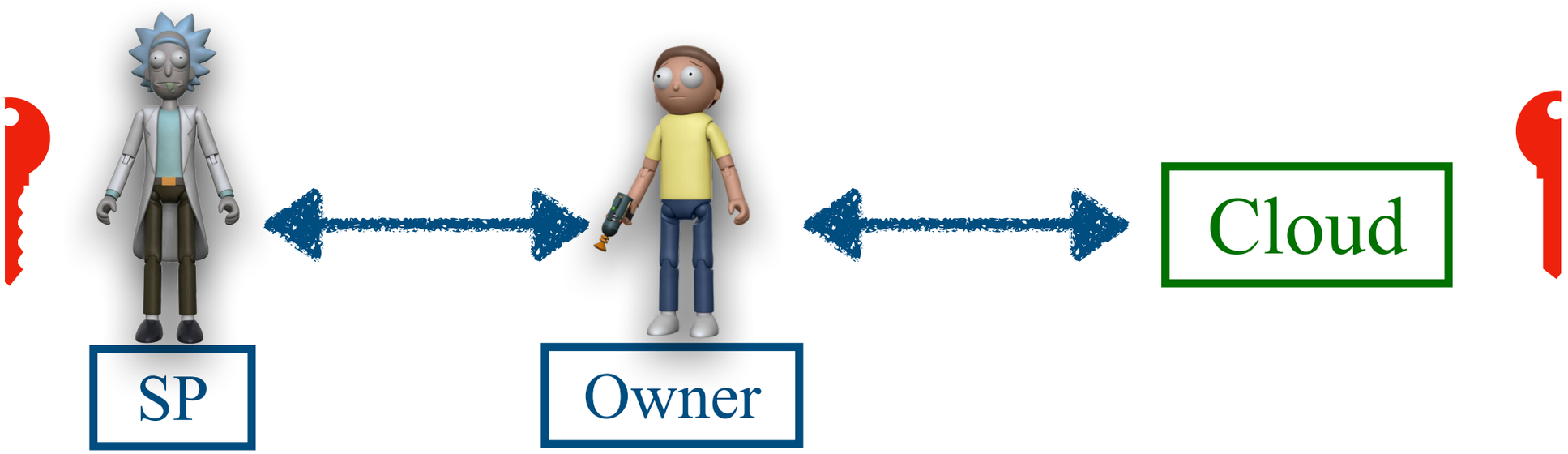
Recovery in Mixed Model{t=n=2}

- Owner self-recovery reduces to classical backup



Recovery in Mixed Model{t=n=2}

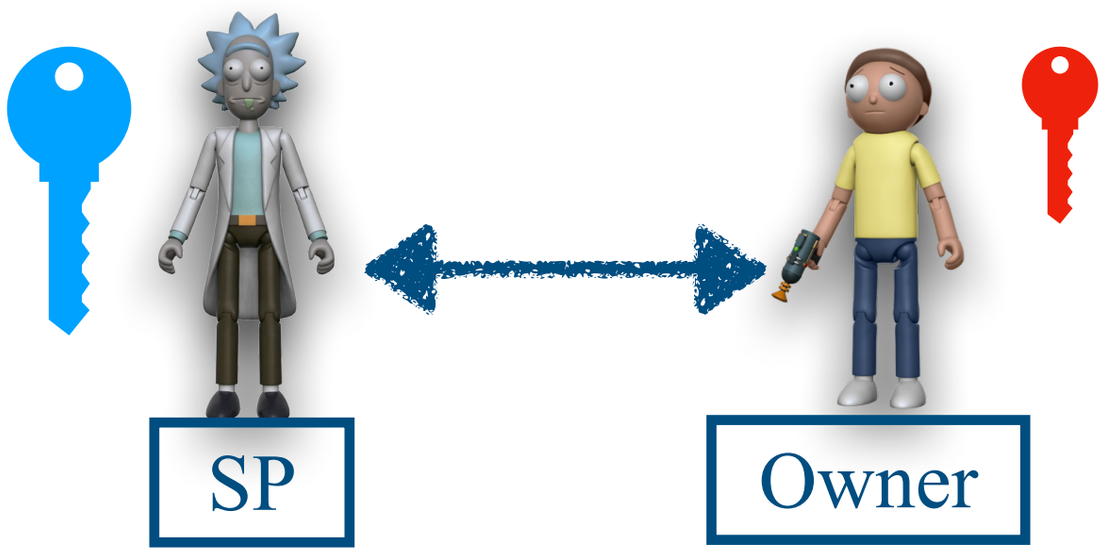
- Owner self-recovery reduces to classical backup
- Assuming Authentication:



KZ

Recovery in Mixed Model{t=n=2}

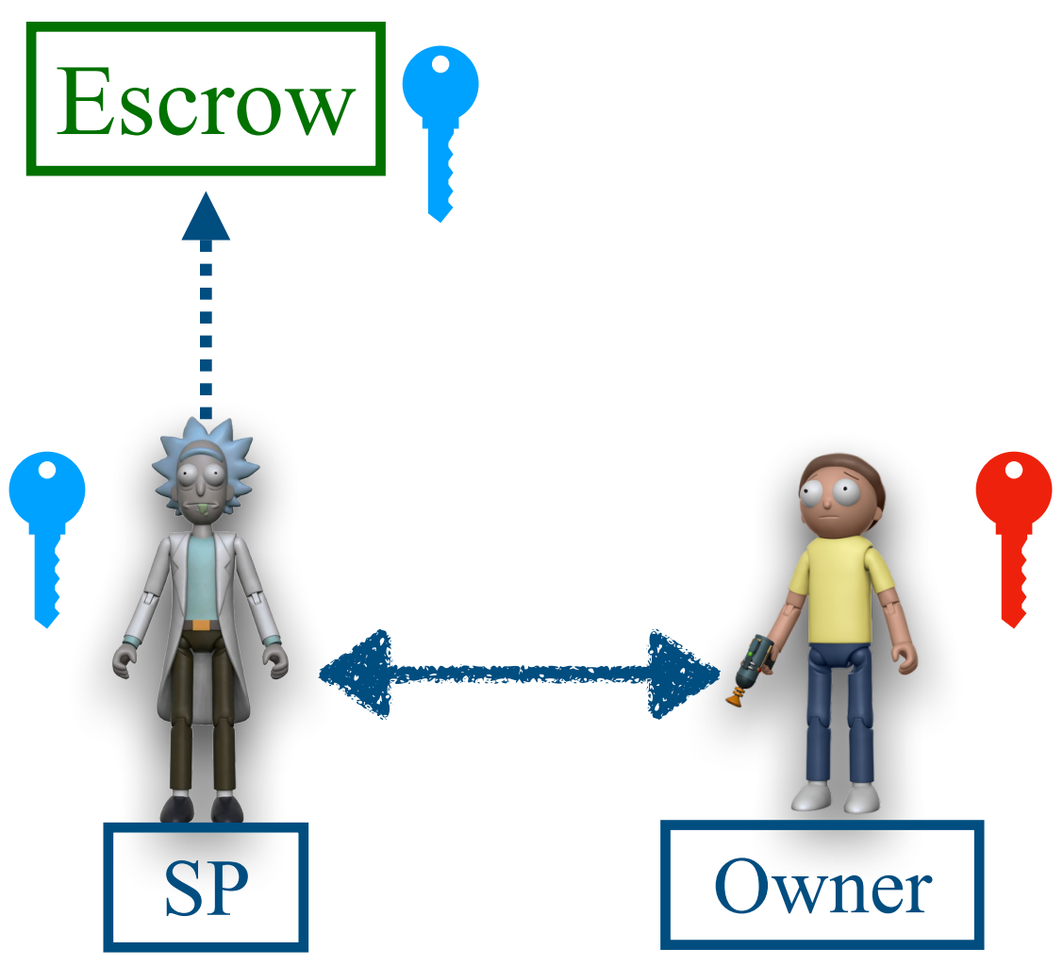
- Recovery in the two party setting can mean:
  - Self recovery: Owner's secret share
  - **Counter party recovery: SP secret share**
    - How can the Owner recover if SP goes offline / hacked / becomes malicious ?



$$sk = f(\ \ ,\ \ )$$

Recovery of Counter Party Secret

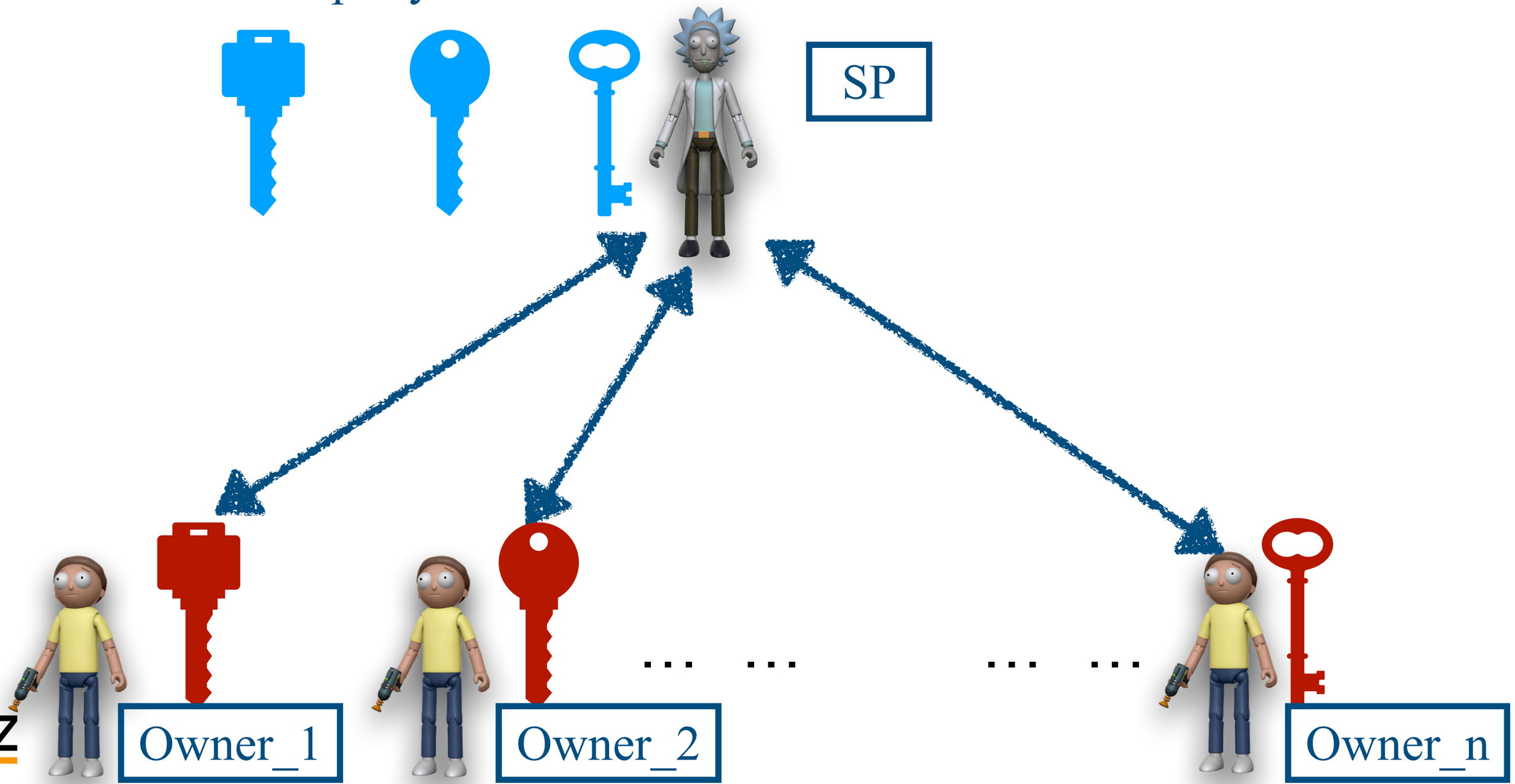
- How can we recover Counter secret share if SP goes offline / hacked / becomes malicious ?
- Under certain assumptions this can be done easily.
  - Escrow service that is triggered to release SP secret share once SP is not sending a life signal for a certain period of time



$$sk = f(\ \ ,\ \ )$$

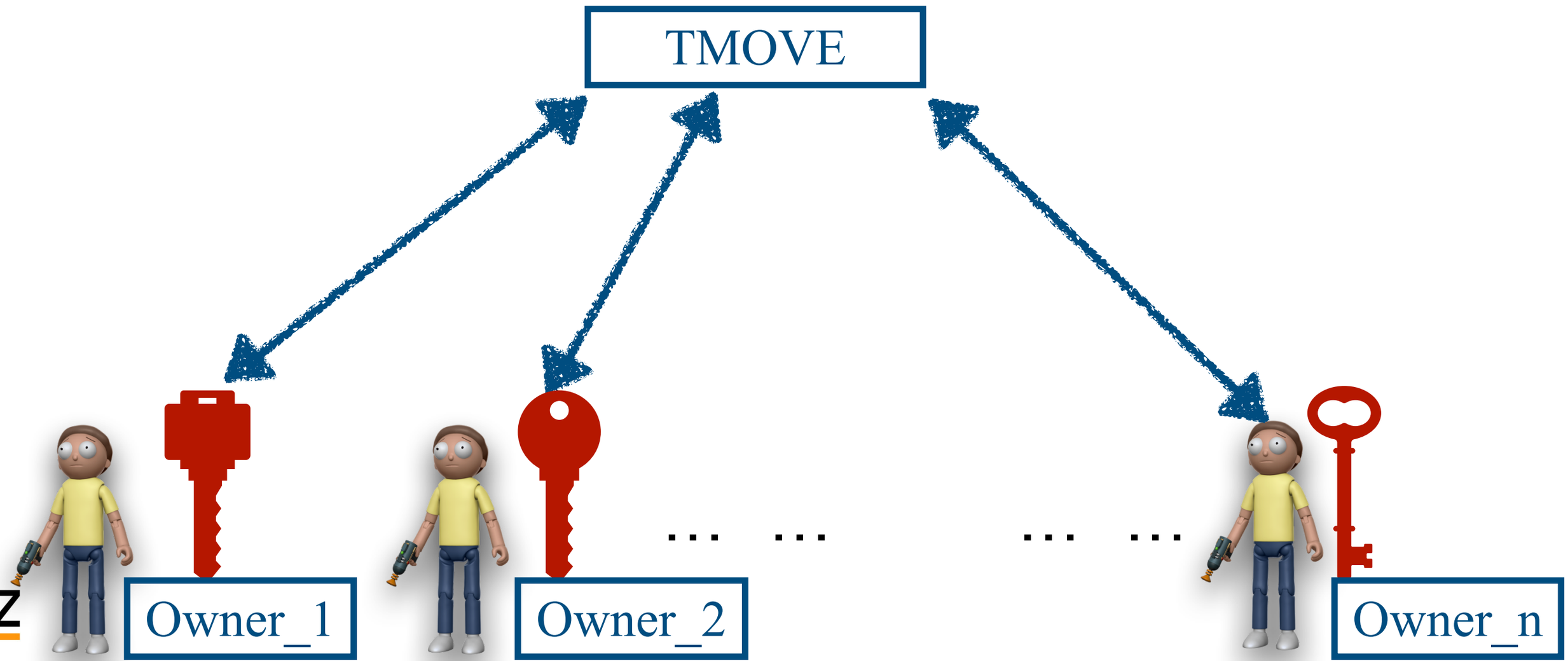
Recovery of Counter Party Secret

- general idea:
  - If enough Owners collaborate, they each get to recover their Counter party secret share at the same time



Recovery of Counter Party Secret

- general idea:
  - If enough Owners collaborate, they each get to recover their Counter party secret share at the same time



Recovery of Counter Party Secret

- general idea:
  - If enough Owners collaborate, they each get to recover their Counter party secret share at the same time

Background: PVSS [S99]

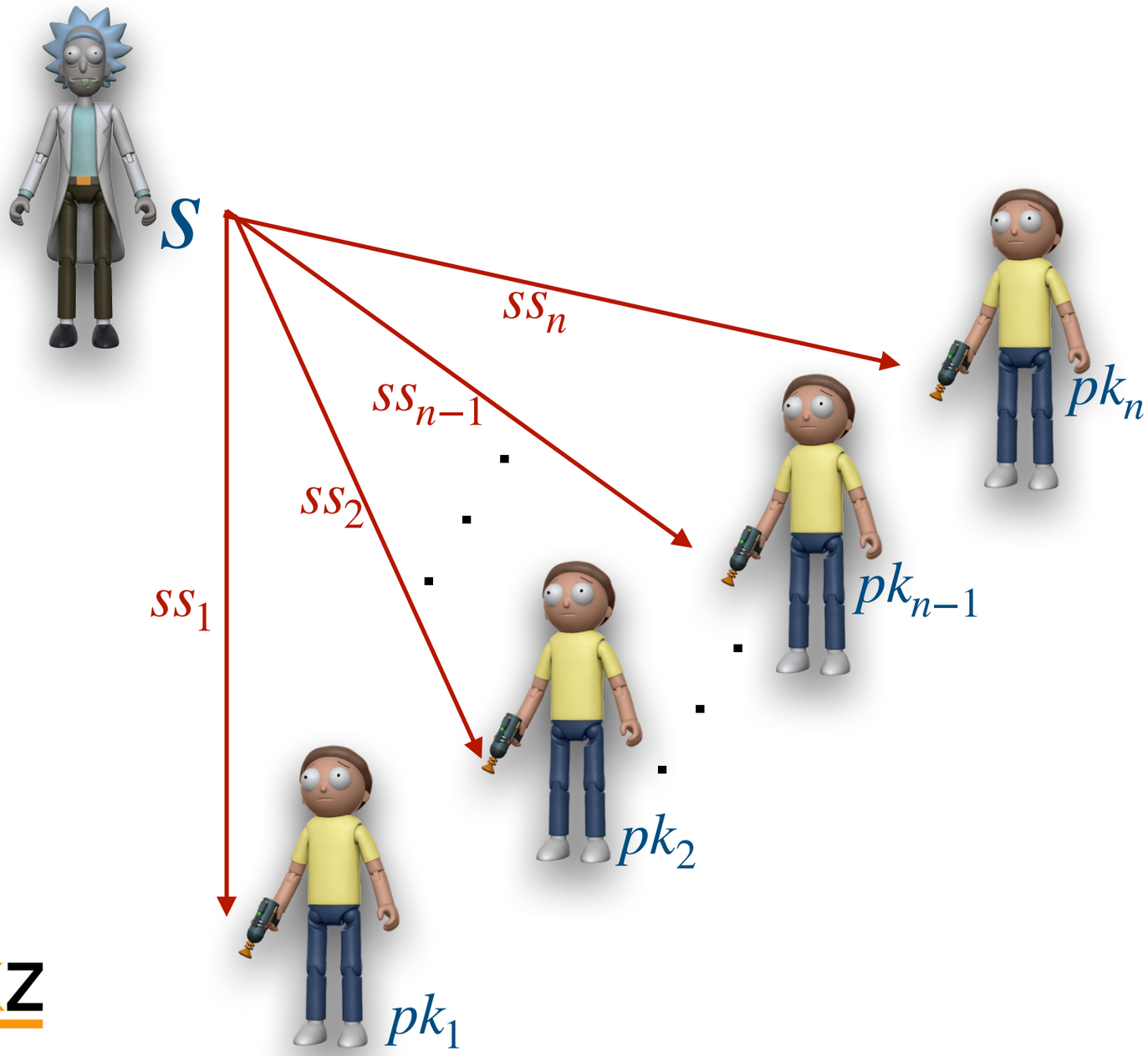
Publicly Verifiable Secret Sharing

Background: PVSS

Publicly Verifiable Secret Sharing

Dealer

Distribution



KZ

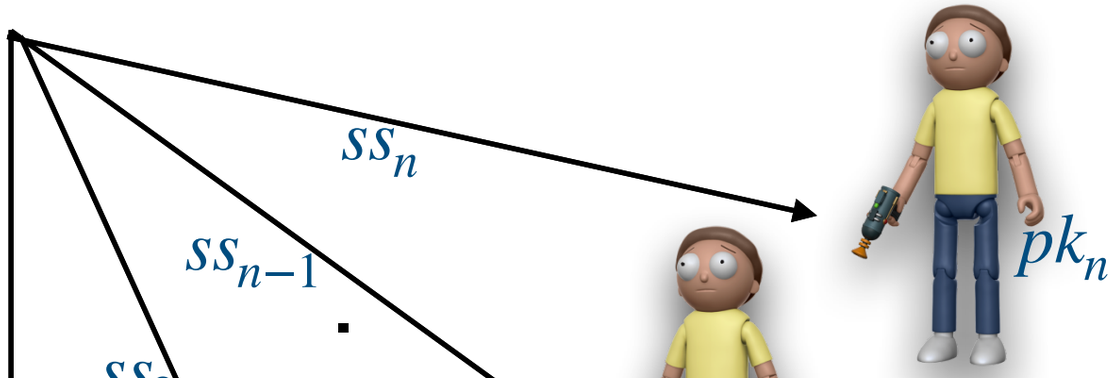
Background: PVSS

Publicly Verifiable Secret Sharing

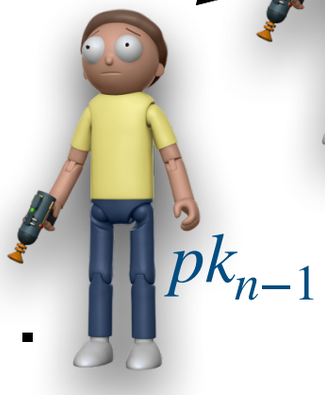
public proofs : A dealer cannot send incorrect shares

Dealer

s



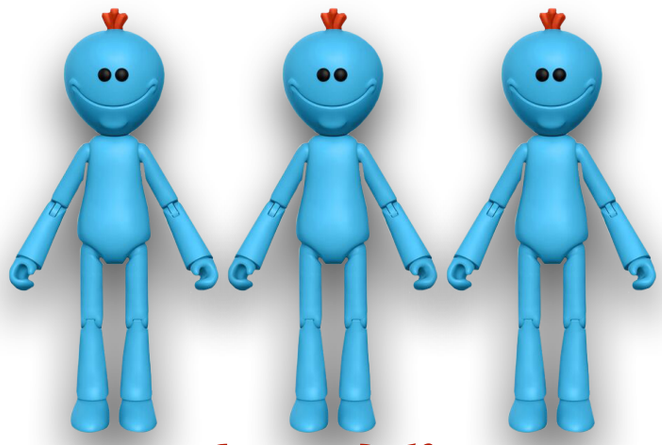
pk_n



pk_{n-1}

pk_2

pk_1



$\{\pi_{d_i}\}_1^n$

KZ

Background: PVSS

Publicly Verifiable Secret Sharing

Dealer

s

Reconstruction: t out of n

pk_t

ss'_t

pk_{t-1}

ss'_{t-1}

pk_2

ss'_2

pk_1

ss'_1

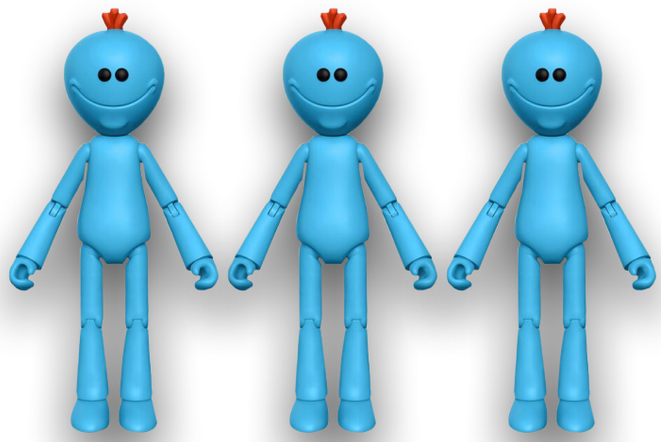
S

KZ

Background: PVSS

Publicly Verifiable Secret Sharing

Public proofs: Participants cannot submit incorrect shares



$\{\pi_{d_i}\}_1^n \{\pi_{r_i}\}_1^t$

pk_t

ss'_n

pk_{t-1}

ss'_{n-1}

Pk_2

ss'_2

pk_1

ss'_1

S

KZ

Background: DLog VE [CS03]

Verifiable Encryption of Discrete Log

Background: DLog VE [CS03]

Verifiable Encryption of Discrete Log

$$\{Gen, Enc, Dec\}$$

$$\{x, \omega\} \in R_{dlog}$$

$$\{sk_m, pk_m\} \leftarrow Gen(1^n)$$

Background: DLog VE [CS03]

Verifiable Encryption of Discrete Log

$\{Gen, Enc, Dec\}$

$\{x, \omega\} \in R_{dlog}$

$c, \pi \leftarrow Enc^{\star}(\omega, pk_m)$

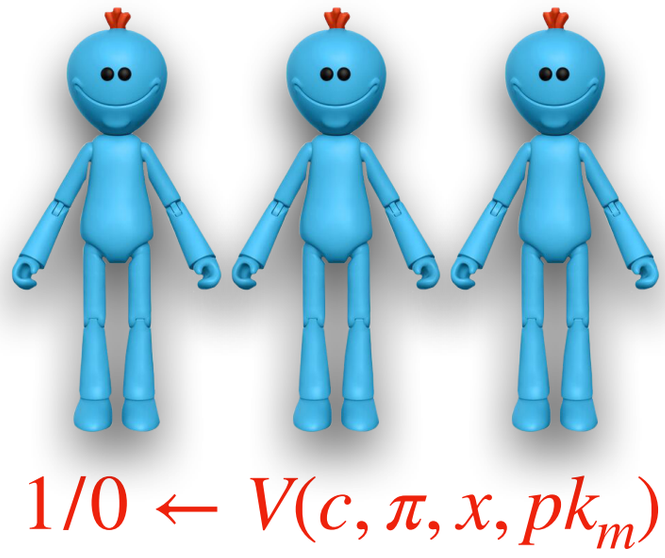
$\{sk_m, pk_m\} \leftarrow Gen(1^n)$

Background: DLog VE [CS03]

Verifiable Encryption of Discrete Log

$\{Gen, Enc, Dec\}$

$\{x, \omega\} \in R_{dlog}$



$c, \pi \leftarrow Enc^{\star}(\omega, pk_m)$

$1/0 \leftarrow V(c, \pi, x, pk_m)$

$\{sk_m, pk_m\} \leftarrow Gen(1^n)$

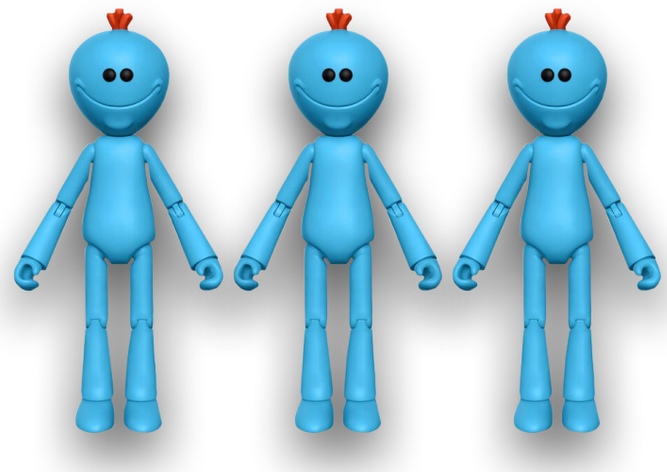
KZ

Background: DLog VE [CS03]

Verifiable Encryption of Discrete Log

$\{Gen, Enc, Dec\}$

$\{x, \omega\} \in R_{dlog}$



$c, \pi \leftarrow Enc^{\star}(\omega, pk_m)$

$1/0 \leftarrow V(c, \pi, x, pk_m)$

$\{sk_m, pk_m\} \leftarrow Gen(1^n)$

$\omega \leftarrow Dec^{\star}(c, sk_m)$

KZ

TVE

Threshold Verifiable Encryption

TVE

Threshold Verifiable Encryption

$$\{Gen, Enc, Dec\}$$

$$\{x, \omega\} \in R_{dlog}$$

$$\{sk_m, pk_m\} \leftarrow Gen(1^n)$$

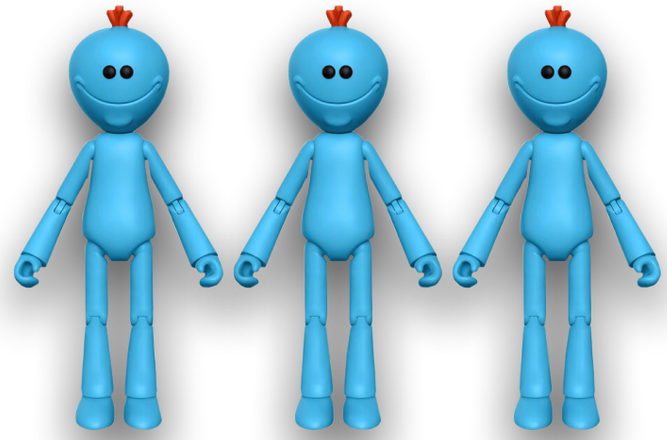
TVE

Threshold Verifiable Encryption

$\{Gen, Enc, Dec\}$

$\{x, \omega\} \in R_{dlog}$

$c, \pi \leftarrow Enc^{\star}(\omega, \{pk_i\}_1^n, pk_m)$



$1/0 \leftarrow V(c, \pi, x, pk_m)$

$\{sk_m, pk_m\} \leftarrow Gen(1^n)$

KZ

TVE

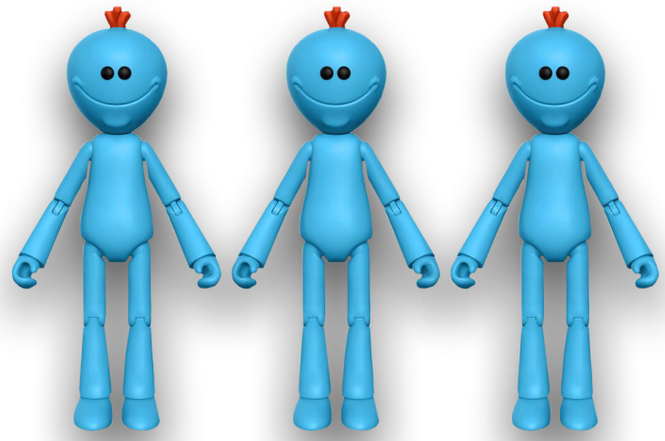
Threshold Verifiable Encryption

$\{Gen, Enc, Dec\}$

$\{x, \omega\} \in R_{dlog}$

$c, \pi \leftarrow Enc^{\star}(\omega, \{pk_i\}_1^n, pk_m)$

PVSS::distribute



$1/0 \leftarrow V(c, \pi, x, pk_m)$

$\{\pi_{d_i}\}_1^n$

$\{sk_m, pk_m\} \leftarrow Gen(1^n)$

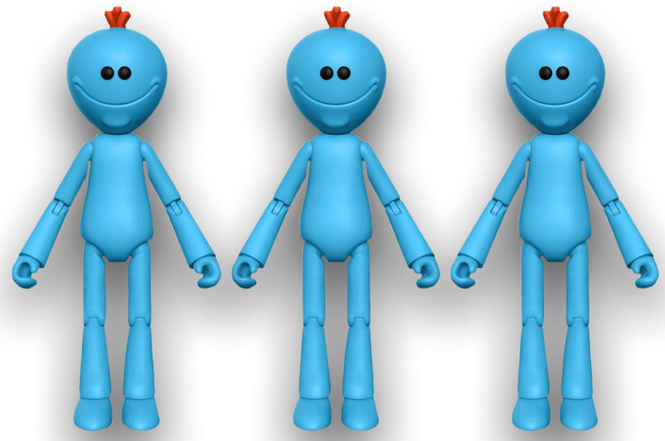
KZ

TVE

Threshold Verifiable Encryption

$\{Gen, Enc, Dec\}$

$\{x, \omega\} \in R_{dlog}$



$1/0 \leftarrow V(c, \pi, x, pk_m)$

$\{\pi_{d_i}\}_1^n \ \{\pi_{r_i}\}_1^t$

$c, \pi \leftarrow Enc^{\star}(\omega, \{pk_i\}_1^n, pk_m)$

pk_1

pk_2

.

.

.

pk_n

PVSS::reconstruct

$\{sk_m, pk_m\} \leftarrow Gen(1^n)$

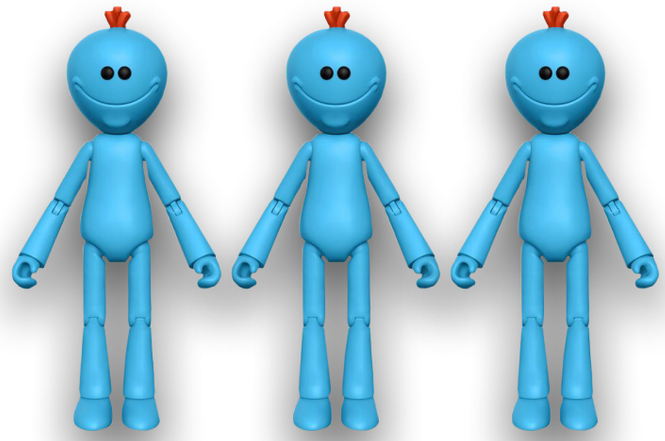
KZ

TVE

Threshold Verifiable Encryption

$\{Gen, Enc, Dec\}$

$\{x, \omega\} \in R_{dlog}$



$1/0 \leftarrow V(c, \pi, x, pk_m)$

$\{\pi_{d_i}\}_1^n \ \{\pi_{r_i}\}_1^t$

$c, \pi \leftarrow Enc^{\star}(\omega, \{pk_i\}_1^n, pk_m)$

pk_1

pk_2

.

.

.

pk_n

$\omega \leftarrow Dec^{\star}(c, sk_m, \{ss_i\}_1^t, Dec)$

$\{sk_m, pk_m\} \leftarrow Gen(1^n)$

KZ

TMOVE

Threshold Multiple Output Verifiable Encryption

TMOVE

Threshold Multiple Output Verifiable Encryption

$$\{Gen, Enc, Dec\}$$

$$\{x, \omega\} \in R_{dlog}$$

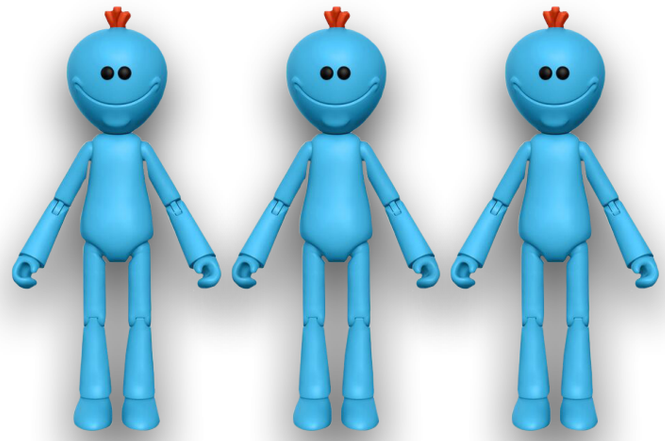
KZ

TMOVE

Threshold Multiple Output Verifiable Encryption

$$\{Gen, Enc, Dec\}$$

$$\{x, \omega\} \in R_{dlog}$$



$$\{c_i\}_1^n, \pi \leftarrow Enc^{\star}(\{\omega_i\}_1^n, \{pk_i\}_1^n)$$

$$1/0 \leftarrow V(\{c_i\}_1^n, \pi, x, \{pk_i\}_1^n)$$

pk_1

pk_2

.

.

.

pk_n

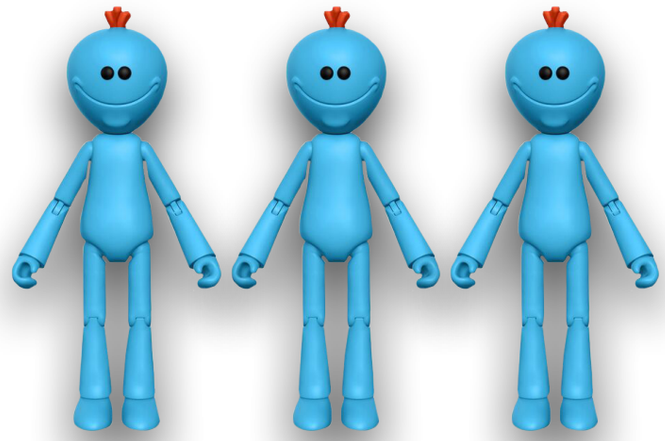
KZ

TMOVE

Threshold Multiple Output Verifiable Encryption

$\{Gen, Enc, Dec\}$

$\{x, \omega\} \in R_{dlog}$



$\{c_i\}_1^n, \pi \leftarrow Enc^{\star}(\{\omega_i\}_1^n, \{pk_i\}_1^n)$

PVSS::distribute

$1/0 \leftarrow V(\{c_i\}_1^n, \pi, x, \{pk_i\}_1^n)$

$\{\{\pi_{d_{ij}}\}_{i=1}^n\}_{j=1}^m$

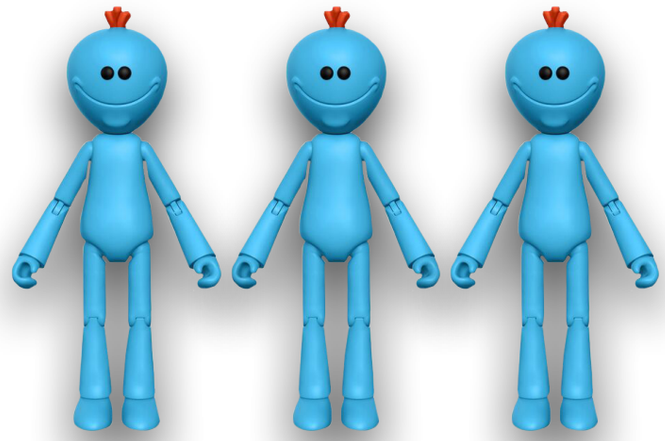
KZ

TMOVE

Threshold Multiple Output Verifiable Encryption

$\{Gen, Enc, Dec\}$

$\{x, \omega\} \in R_{dlog}$



$\{c_i\}_1^n, \pi \leftarrow Enc^{\star}(\{\omega_i\}_1^n, \{pk_i\}_1^n)$

$1/0 \leftarrow V(\{c_i\}_1^n, \pi, x, \{pk_i\}_1^n)$

$\{\{\pi_{d_{ij}}\}_{i=1}^{n}\}_{j=1}^{m}$

$\{\{\pi_{r_{ij}}\}_{i=1}^{t_j}\}_{j=1}^{m}$

PVSS::reconstruct

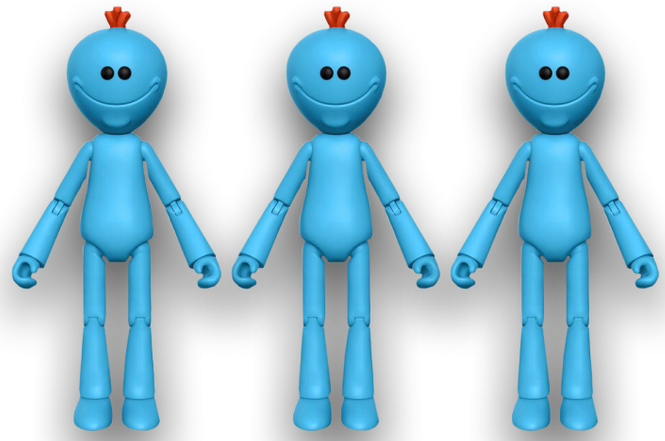
KZ

TMOVE

Threshold Multiple Output Verifiable Encryption

$$\{Gen, Enc, Dec\}$$

$$\{x, \omega\} \in R_{dlog}$$



$$\{c_i\}_1^n, \pi \leftarrow Enc^{\star}(\{\omega_i\}_1^n, \{pk_i\}_1^n)$$

$$1/0 \leftarrow V(\{c_i\}_1^n, \pi, x, \{pk_i\}_1^n)$$

$$\left\{\left\{\pi_{d_{ij}}\right\}_{i=1}^{n}\right\}_{j=1}^{m}$$

$$\left\{\left\{\pi_{r_{ij}}\right\}_{i=1}^{t_j}\right\}_{j=1}^{m}$$

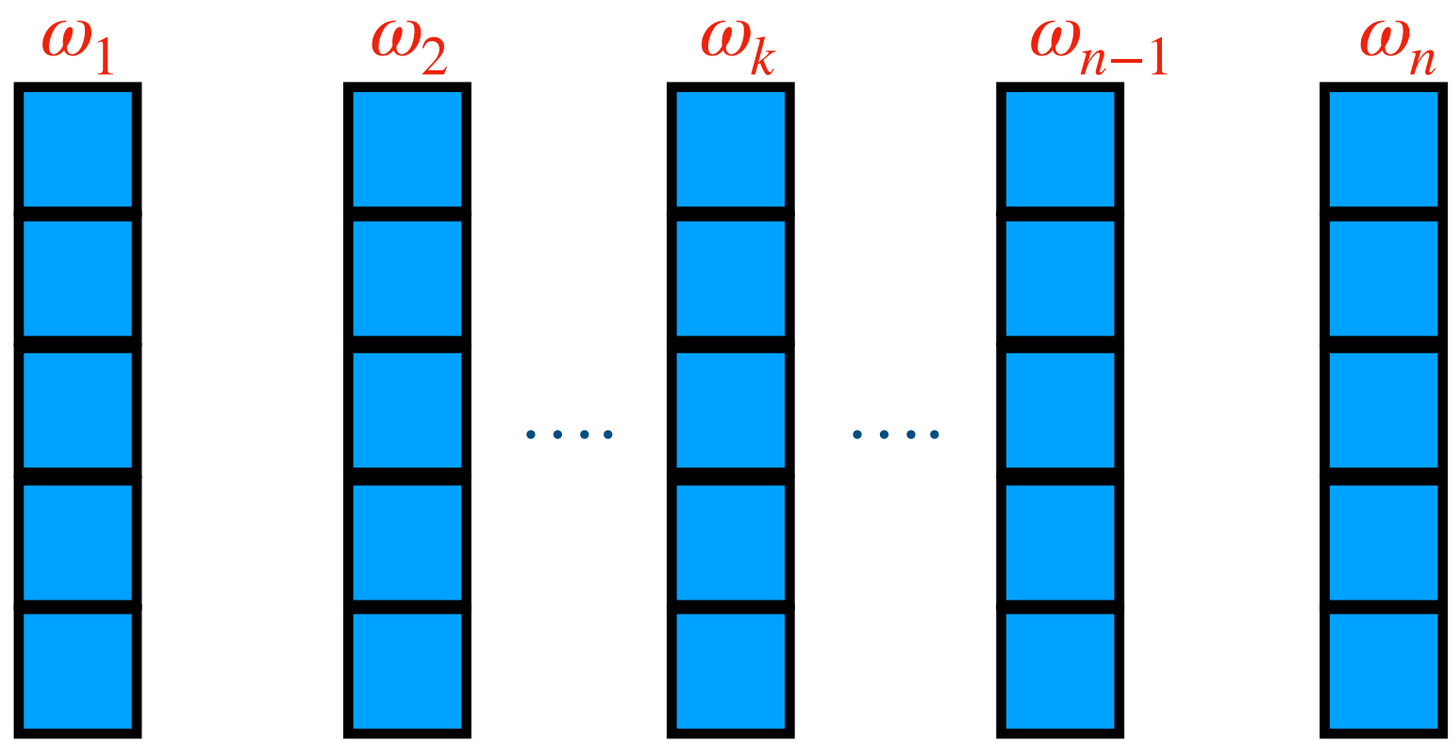
$$\omega_k \leftarrow Dec^{\star}(c_k, sk_k, \{\{ss_i\}_{i=1}^{t_j}\}_{j=1}^{m})$$

KZ

TMOVE Properties

* TVE per party

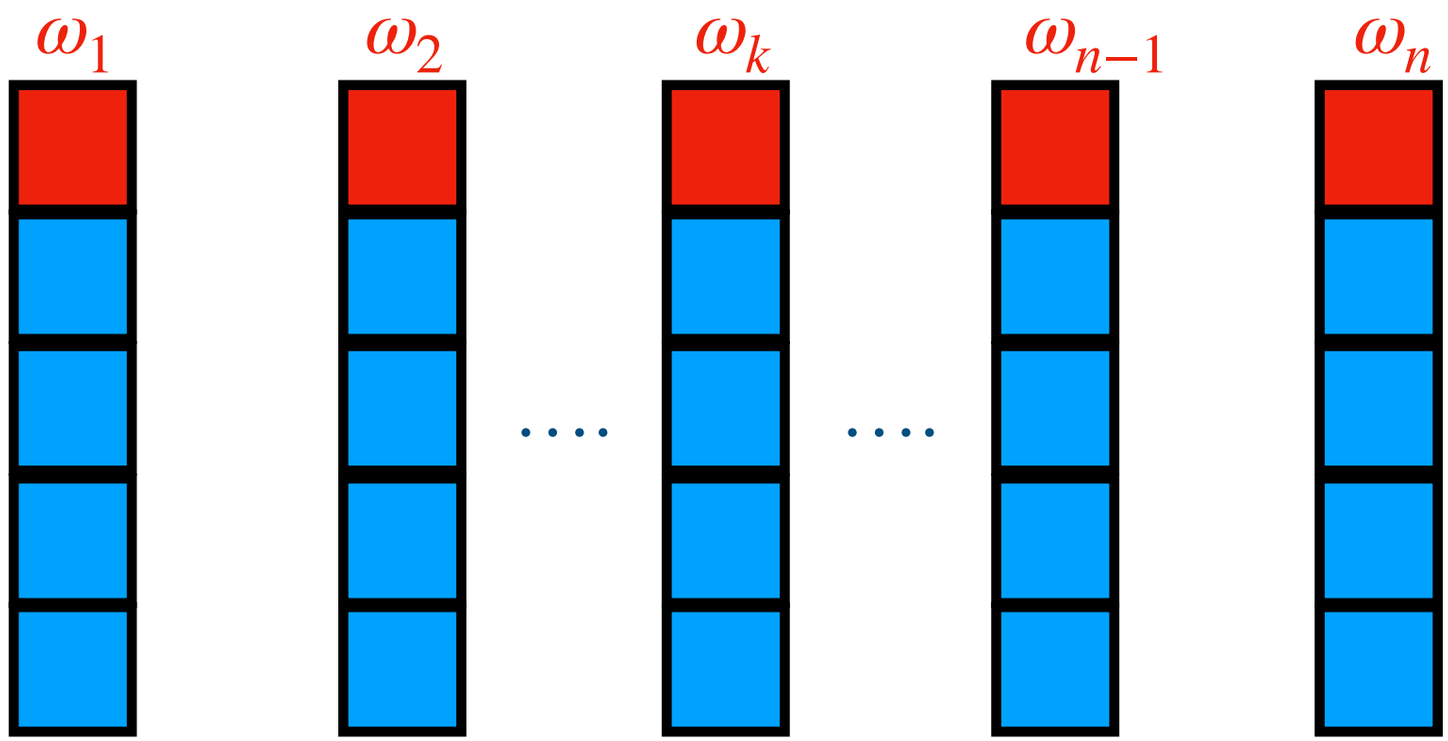
* Gradual release (up to one segment)



TMOVE Properties

* TVE per party

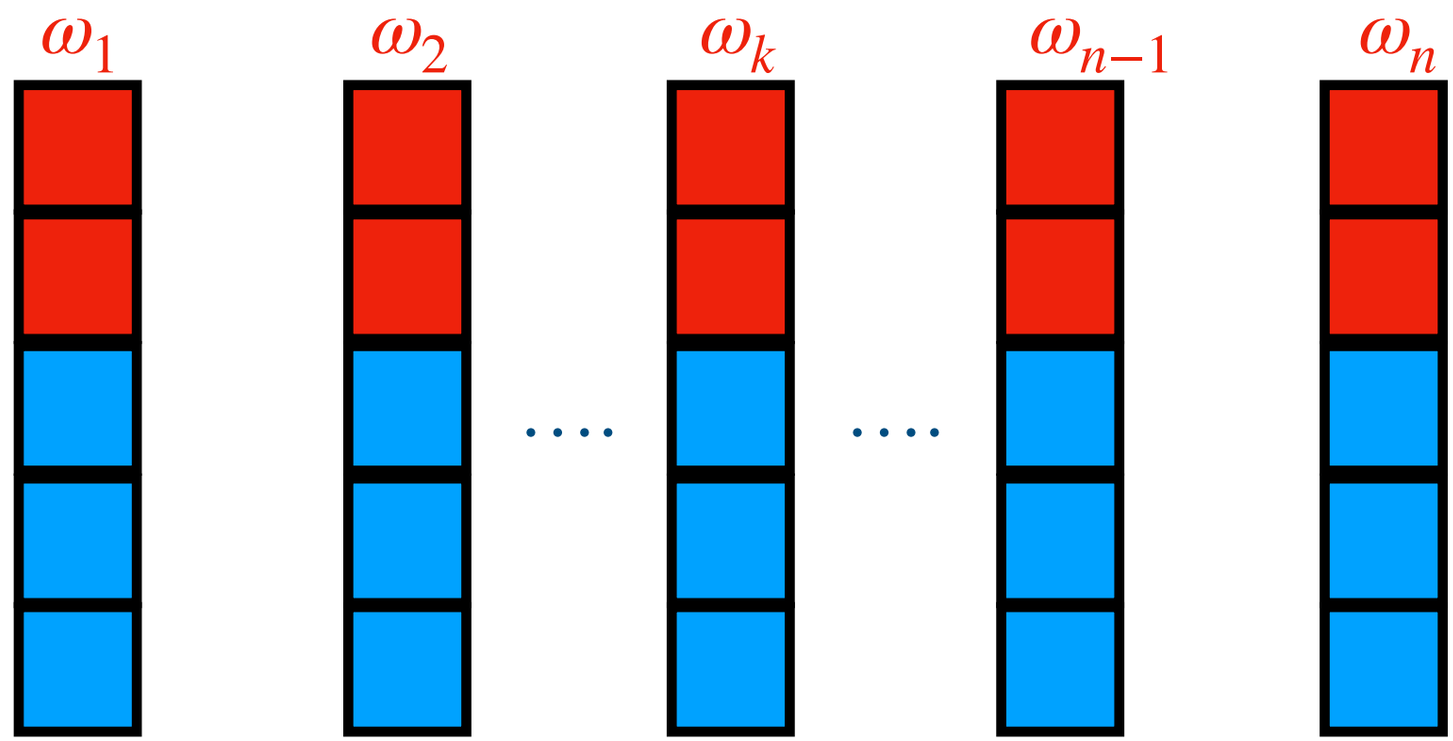
* Gradual release (up to one segment)



TMOVE Properties

* TVE per party

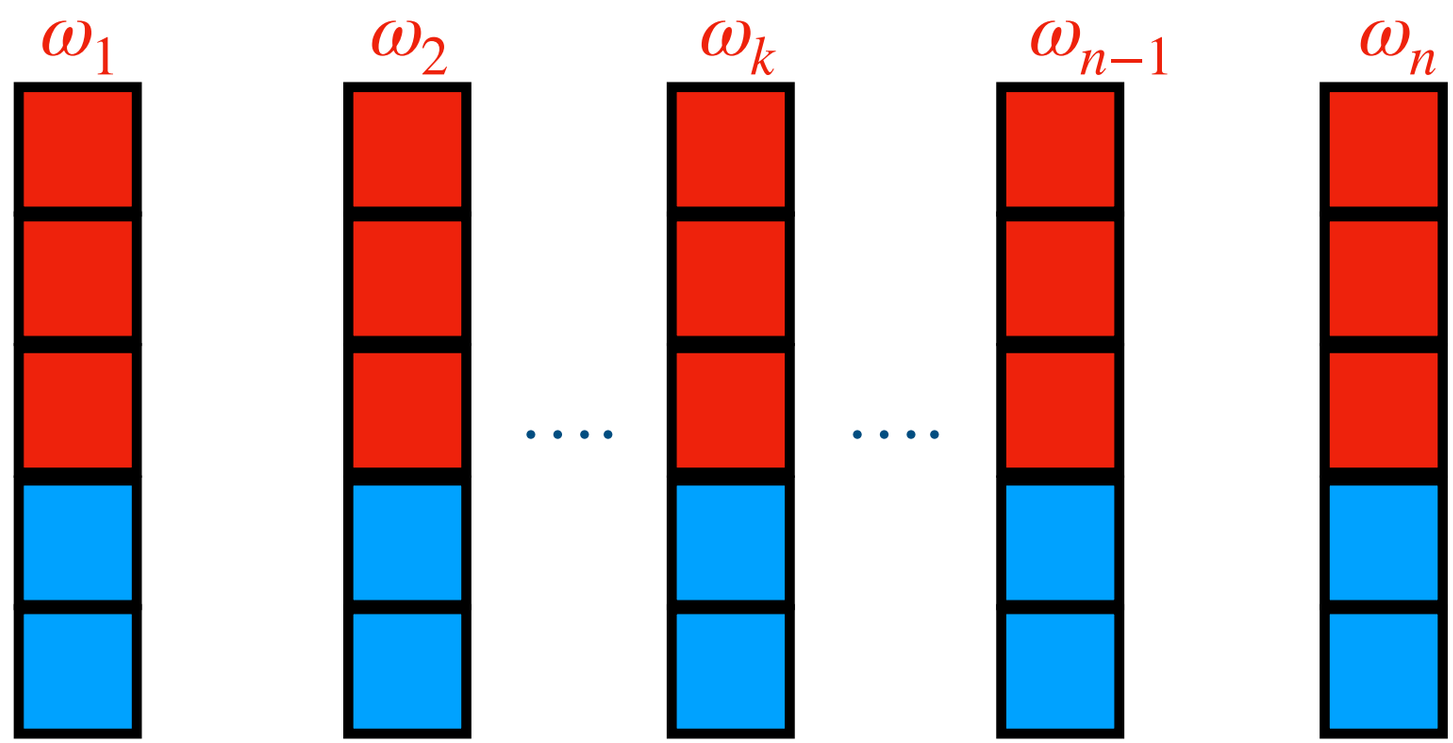
* Gradual release (up to one segment)



TMOVE Properties

* TVE per party

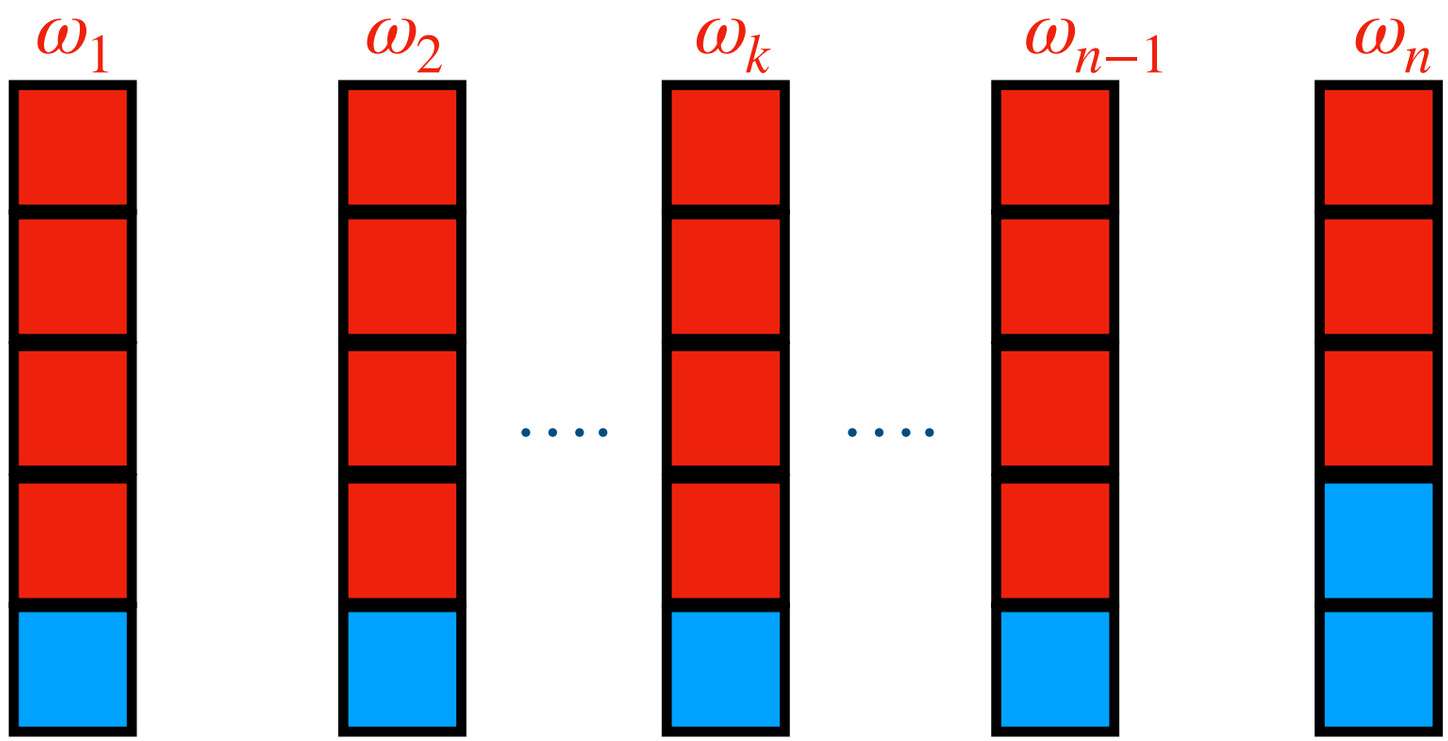
* Gradual release (up to one segment)



TMOVE Properties

* TVE per party

* Gradual release (up to one segment)



TMOVE Instantiation

* Additive Homomorphic Encryption scheme: ElGamal “in the exponent” (homomorphic ElGamal) :

$$Enc_Y(\omega) = \{C_1, C_2\} = \{\ \omega G\ +\ \ rY, rG\}$$

Additively-Homomorphism:

$$\begin{array}{rl} & (B_1, B_2) = \{\omega G + rY, rG\} \\ + & (C_1, C_2) = \{aG + sY, sG\} \\ \hline = & (D_1, D_2) = \{(a+\omega)G + (r+s)Y, (r+s)G\} \end{array}$$

TMOVE Instantiation

* Additive Homomorphic Encryption scheme: ElGamal "in the exponent" (homomorphic ElGamal):

$$Enc_Y(\omega) = \{C_1, C_2\} = \{\ \omega G\ +\ \ rY, rG\}$$

* We do the following for party i and PVSS secret α_j :

$$C_1[i,j] = [\omega_i]_j G + \alpha_j pk_i$$

TMOVE Instantiation

* Additive Homomorphic Encryption scheme: ElGamal "in the exponent" (homomorphic ElGamal) :

$$Enc_Y(\omega) = \{C_1, C_2\} = \{\ \omega G\ +\ \ rY, rG\}$$

* We do the following for party i and PVSS secret α_j :

$$C_1[i,j] = [\omega_i]_j G + \alpha_j pk_i$$

* Verifiable Encryption: ZK proof that $C_1[i,j]$ is an encryption of a small witness segment under public key of party i with randomness equal to α_j

TMOVE Instantiation

* Additive Homomorphic Encryption scheme: ElGamal "in the exponent" (homomorphic ElGamal) :

$$Enc_Y(\omega) = \{C_1, C_2\} = \{\ \omega G\ +\ \ rY, rG\}$$

* We do the following for party i and PVSS secret α_j :

$$C_1[i,j] = [\omega_i]_j G + \alpha_j pk_i$$

* Verifiable Encryption: ZK proof that $C_1[i,j]$ is an encryption of a small witness segment under public key of party i with randomness equal to α_j

* Gradual release: α_j are reconstructed one at a time such that at any given moment the difference between parties is no more than one encrypted segment

Verifiable Social Recovery via TMOVE

Service Provider

Owner i

Verifiable Social Recovery via TMOVE

Service Provider

Owner i

2P keyGen

s_r

s_m

$sk = f(s_r, s_m)$

Verifiable Social Recovery via TMOVE

Service Provider

Owner i

s_r

2P keyGen

s_m

TMOVE::encrypt(s_r)

TMOVE::distribute::party(i)

Verifiable Social Recovery via TMOVE/ part2

Verifiable Social Recovery via TMOVE/ part2

TMOVE::reconstruct

Verifiable Social Recovery via TMOVE/ part2

$$sk = f(s_r, s_m)$$

TMOVE::decrypt::party(i)

Mixed Model {t=n=2}

* Roles
  - *Owner x 1*
  - *Service Providers x 1*

* System Requirements
  - ☑ *No single point of failure*
  - ☑ *Move of assets (signing) cannot happen without Owner approval*
  - *Recovery is possible at all times* ?

KZ

Mixed Model {t=n=2}

* Roles
  - *Owner x 1*
  - *Service Providers x 1*

* System Requirements
  - ☑ *No single point of failure*
  - ☑ *Move of assets (signing) cannot happen without Owner approval*
  - ☑ *Recovery is possible at all times*

The Journey

Trust that other users in the same situation will act rationally

Verifiable Social Recovery

1 owner, 1 sp.
No single point of failure

Mixed Model

KMS

Key management is hard.

TMOVE

Threshold multiple output verifiable encryption

https://github.com/KZen-networks

Practical Considerations

- One SP can handle millions of Owners
- Owners can join the service Asynchronously
- Owners of the same SP must have similar stake in the system
- PKI: Owners of the same SP must know each other public key (blockchain pk's are good)