Threshold Scriptless Scripts

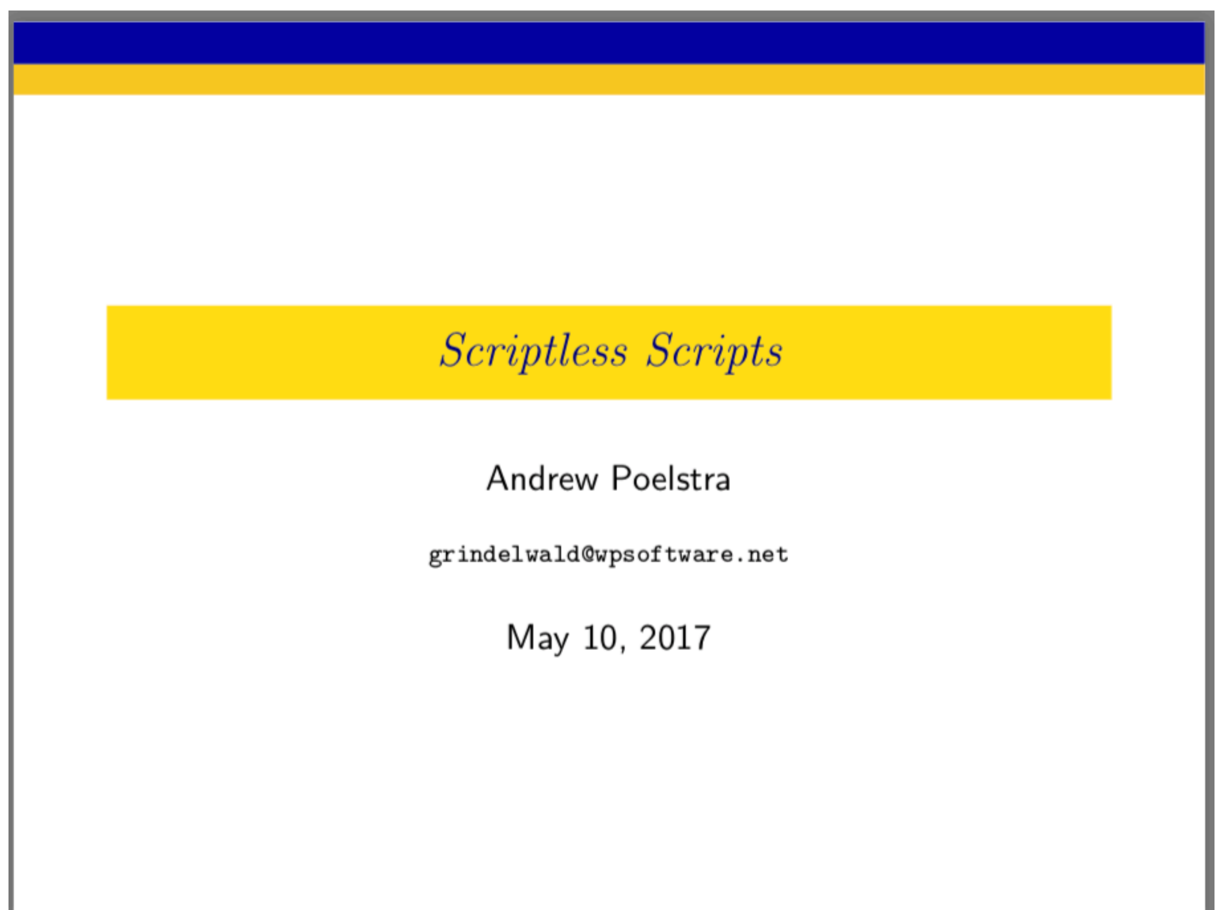
Omer Shlomovits

KZen Research

Scriptless Scripts

"Magicking digital signatures so that they can only be created by faithful execution of a smart contract".

Andrew Poelstra



In This Talk…

- Intro to Schnorr Scriptless Scripts (SSS)

- The road to ECDSA Scriptless Scripts (ESS)
  - Rebuttal ECDSA known security issues
  - Discussing 2P-ECDSA as main tool
  - Expending the Tool Box with Threshold ECDSA

- Experimenting with ESS

Schnorr Signature

EC public parameters : q, G

- Choose random k
- Compute $R = k \cdot G$

- Compute $s = k + H(R, P, m) \cdot x \bmod q$
  where x is the private key, $R = x \cdot G$
- Output (R, s)

The Shtik

* Adaptor Signature: Main building block is a tweak to Schnorr signature (R,s)

$$s' = s + t$$

* Schnorr Sig
* Adaptor (t, T)

shtik

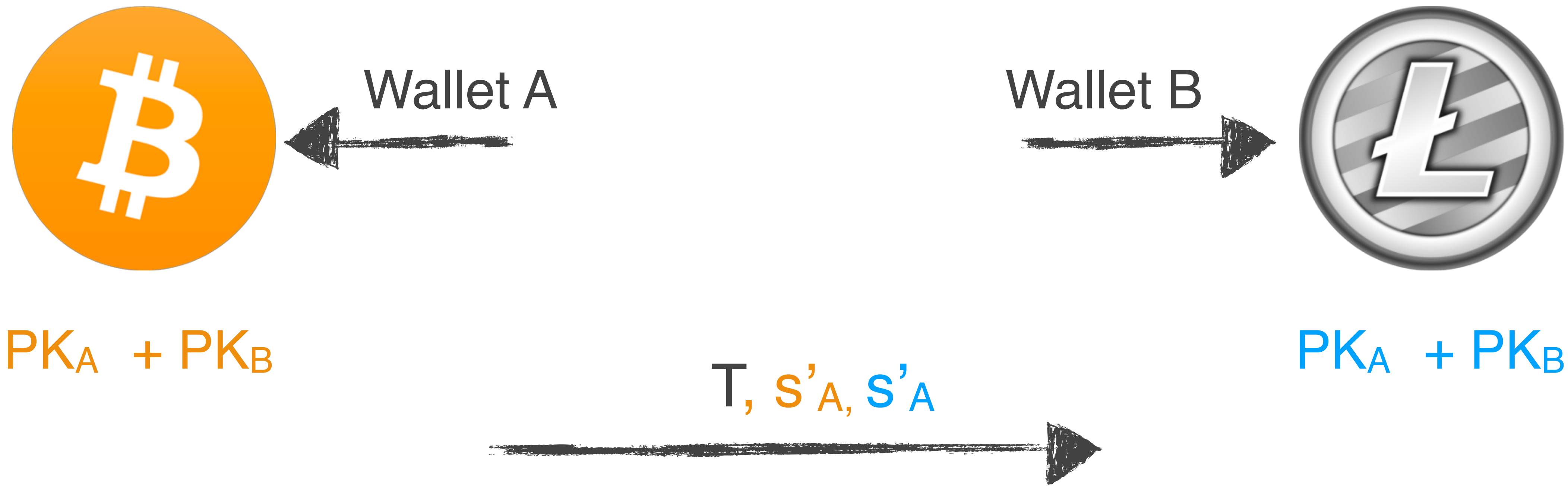
(Yiddish) a devious trick;

ElementsProject/scriptless-scripts

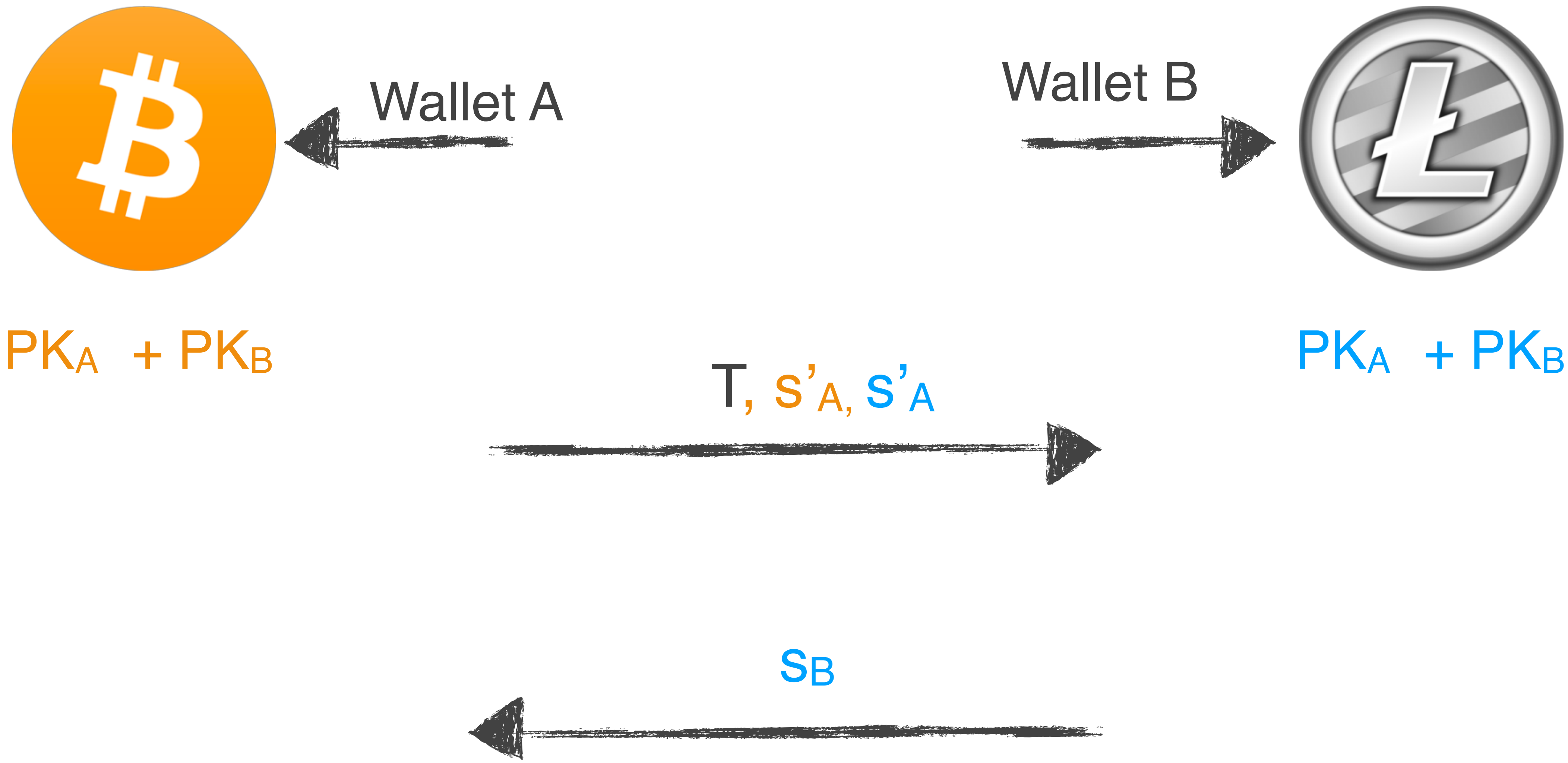
SS Atomic Swap

$PK_A + PK_B$ $PK_A + PK_B$

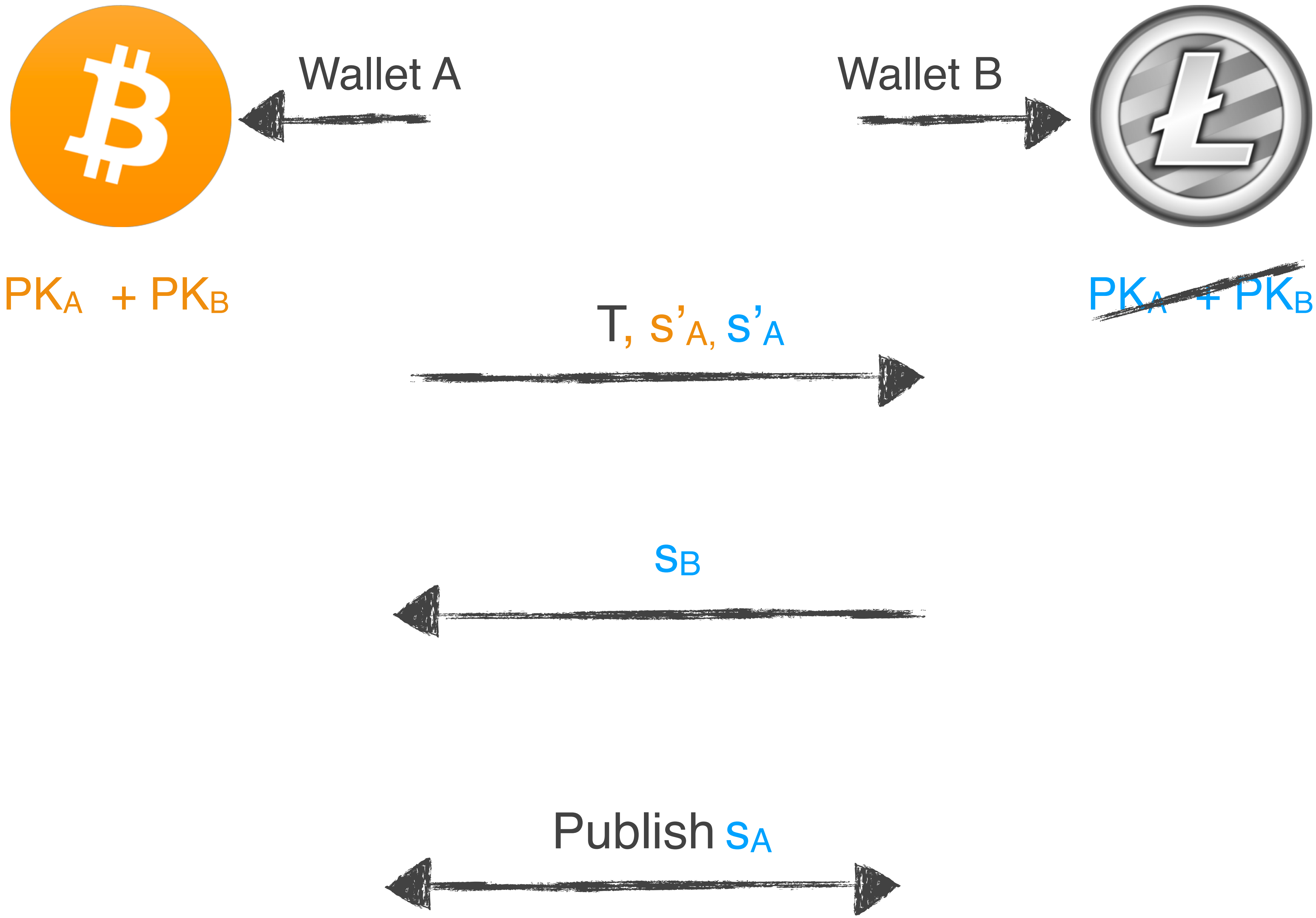
SS Atomic Swap



SS Atomic Swap



SS Atomic Swap



SS Atomic Swap

Wallet A

Wallet B

$\cancel{PK_A + PK_B}$

$\cancel{PK_A + PK_B}$

T, s'_A, s'_A

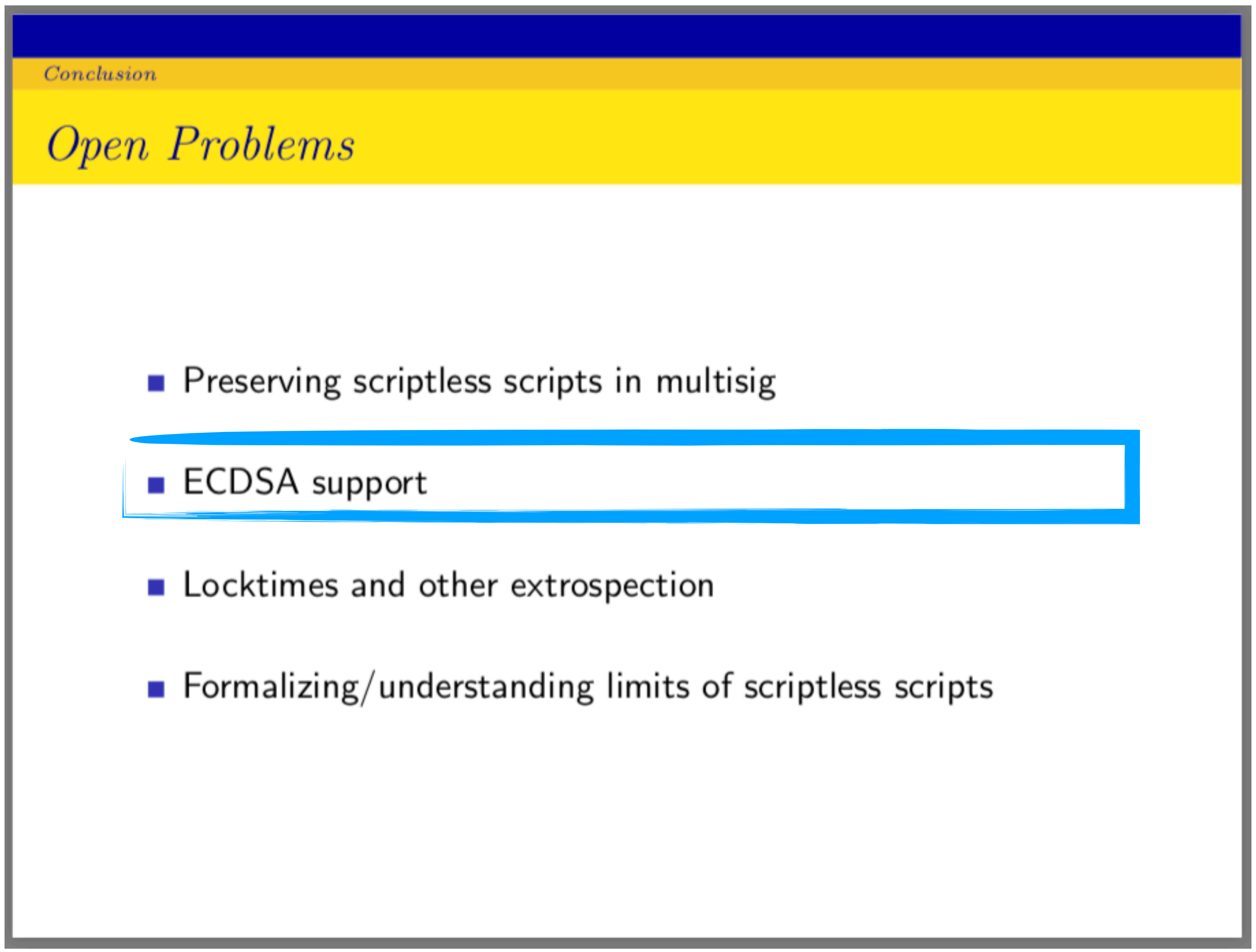
s_B

Publish s_A

$t = s'_A - s_A$

$s_A = s'_A - t$

Scriptless Scripts



[ElementsProject/scriptless-scripts](https://github.com/ElementsProject/scriptless-scripts)

ECDSA SS - The Hard Questions

- Why we needed Schnorr in the first place?
- What are the challenges in using ECDSA for SS

Why Schnorr ?

EC public parameters : q,G

ECDSA

- Choose random k
- Compute $R = k \cdot G$
- Compute $r = r_x \bmod q$ where $R = (r_x, r_y)$
- Compute $s = k^{-1} \cdot (H(m) + r \cdot x) \bmod q$ where x is the private key
- Output (r,s)

Schnorr

- Choose random k
- Compute $R = k \cdot G$
- Compute $s = k + H(R, P, m) \cdot x \bmod q$ where x is the private key, $R = x \cdot G$
- Output (R,s)

Why Schnorr #2

ECDSA:

- No security proof
- Malleable
- Not linear

EC-Schnorr:

- Provably secure under ROMDL
- Provably non-malleable
- Linearity!

CIW19 Keynote

By Pieter Wuille
March 20, 2019

Why Schnorr #2

ECDSA:

- No security proof
- Malleable
- Not linear

EC-Schnorr:

- Provably secure under ROMDL
- Provably non-malleable
- Linearity!

CIW19 Keynote

PW By Pieter Wuille
March 20, 2019

ECDSA is a NO-GO?

KZ

ECDSA: No security proof?

ECDSA: No security proof?

The Security of DSA and ECDSA

Bypassing the Standard Elliptic Curve Certification Scheme

Serge Vaudenay

Swiss Federal Institute of Technology (EPFL)
`Serge.Vaudenay@epfl.ch`

- 2003, Generic Group Model

ECDSA: No security proof?

On the Provable Security of (EC)DSA Signatures

Manuel Fersch
manuel.fersch@rub.de

Eike Kiltz
eike.kiltz@rub.de

Bertram Poettering
bertram.poettering@rub.de

Horst Görtz Institute for IT Security
Ruhr University Bochum, Germany

- 2016, Bijective Random Oracle (BRO) model

ECDSA: No security proof?

- ECDSA is one of the widely used signature schemes: TLS, PGP, S/MIME, multiple cryptocurrencies etc..
- ECDSA is widely standardized: IEEE P1363, ANSI X9.62, FIPS 186-4
- It was subject to massive cryptanalytic efforts with zero known attacks

Implementations [edit]

Below is a list of cryptographic libraries that provide support for ECDSA:

- Botan
- Bouncy Castle
- cryptlib
- Crypto++
- libgcrypt
- OpenSSL
- wolfCrypt
- mbed TLS

ECDSA: Malleable ?

- Known malleability:
  - (r,s) and (r, q - s) are both valid signatures on message m

ECDSA: Malleable ?

- Known malleability:
  - (r,s) and (r, $q - s$) are both valid signatures on message m

- Define ECDSA’ as ECDSA that fix this malleability
  - ECDSA’ is strong unforgeable

ECDSA: Malleable ?

- Known malleability:
  - (r,s) and (r, q - s) are both valid signatures on message m

- Define ECDSA’ as ECDSA that fix this malleability
  - ECDSA’ is strong unforgeable

- BIP62 (WIP):

Low S values in signatures

The value S in signatures must be between 0x1 and 0x7FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 5D576E73 57A4501D DFE92F46 681B20A0 (inclusive). If S is too high, simply replace it by S' = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141 - S.

ECDSA: Malleable ?

- Known malleability:
  - (r,s) and (r, q - s) are both valid signatures on message m

- Define ECDSA’ as ECDSA that fix this malleability
  - ECDSA’ is strong unforgeable

- BIP62 (WIP):

Low S values in signatures

The value S in signatures must be between 0x1 and 0x7FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 5D576E73 57A4501D DFE92F46 681B20A0 (inclusive). If S is too high, simply replace it by S' = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141 - S.

- Segwit

Schnorr: Malleable ?

Q:

A:

scheme	public key	first component	second component	sign. size
[Sc91]	$-d \times G$	$H(Q, M)$	$k + d\ h$	$b + 2b$
EC-SDSA	$-d \times G$	$H(Q_x \parallel Q_y \parallel M)$	$k + d\ h$	$2b + 2b$
EC-SDSA-opt	$-d \times G$	$H(Q_x \parallel M)$	$k + d\ h$	$2b + 2b$
EC-FSDSA	$-d \times G$	$Q_x \parallel Q_y$	$k + d\ H(Q_x \parallel Q_y \parallel M)$	$4b + 2b$
EC-Schnorr	$d \times G$	$H(M \parallel Q_x)$	$k - d\ h$	$2b + 2b$
libsecp256k1	$d \times G$	Q_x	$k - d\ H(Q_x \parallel M)$	$2b + 2b$

KZ

Linearity

- Valid point!
- Deal breaker?

Linearity

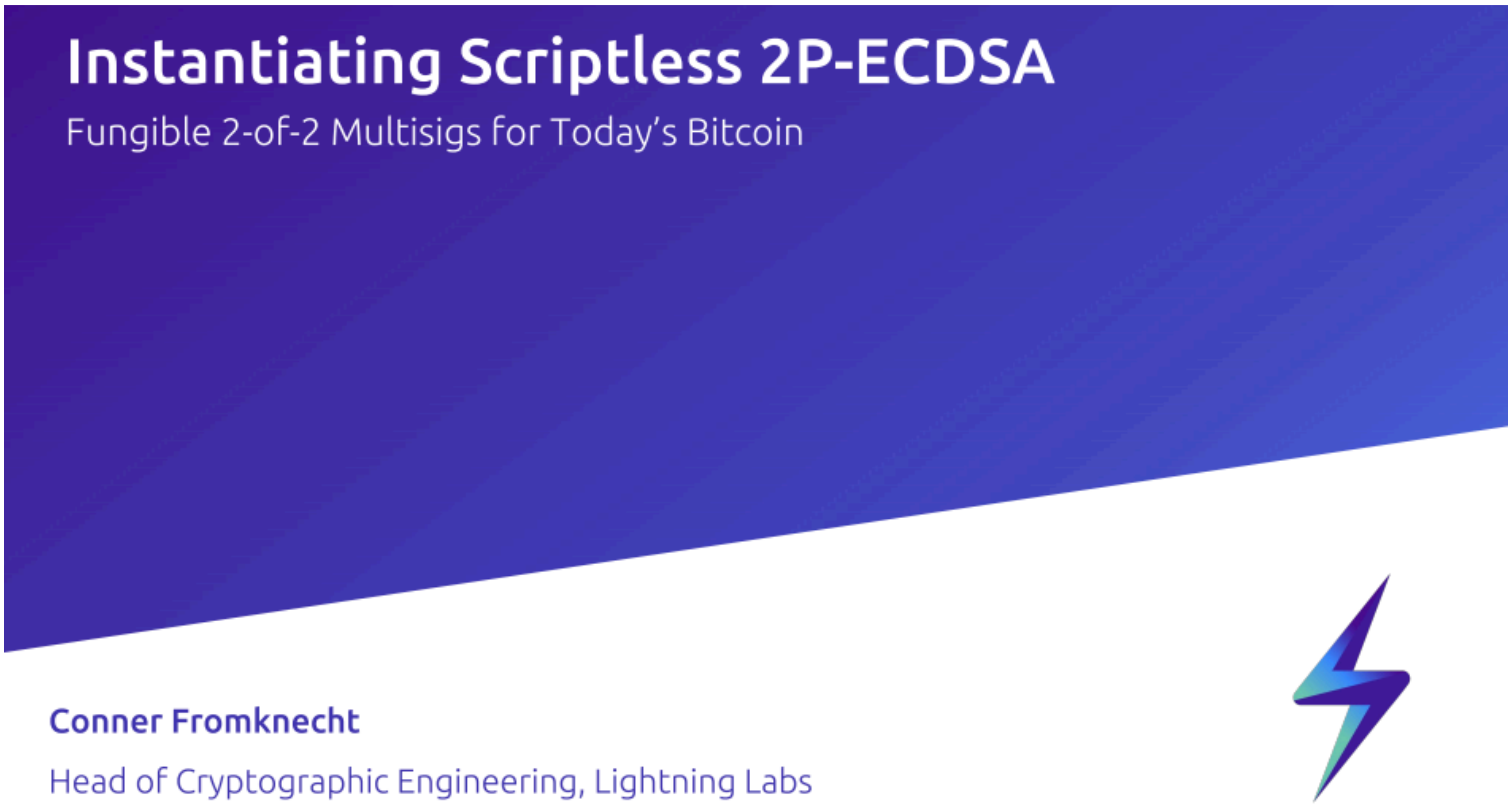
- Valid point!
- Deal breaker?

- Given the non-linearity of ECDSA, Are ECDSA Scriptless Scripts
  - Possible ?
  - Possible but Inefficient ?
  - Possible but with compromise on Security ?

Linearity: Observation

- Threshold ECDSA has struggled with the same problem.
- Not surprisingly
  - Existing works for ECDSA-SS are based on threshold ECDSA protocol

Scaling Bitcoin 2018 #1



Scaling Bitcoin 2018 #2

Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability

Giulio Malavolta*§, Pedro Moreno-Sanchez*¶†, Clara Schneidewind†, Aniket Kate‡, Matteo Maffei†
§Friedrich-Alexander-University Erlangen-Nürnberg, †TU Wien, ‡ Purdue University

Scaling Bitcoin 2018 #2

Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability

Giulio Malavolta[*§], Pedro Moreno-Sanchez[*¶†], Clara Schneidewind[†], Aniket Kate[‡], Matteo Maffei[†]
[§]Friedrich-Alexander-University Erlangen-Nürnberg, [†]TU Wien, [‡] Purdue University

Scriptless Scripts (SS-Schnorr)

- Technique originally proposed by A. Poelstra
- “Encode” payment condition within the Schnorr signatures
- Unfortunately, Schnorr is not used yet in many cryptocurrencies
- In our work:
  - formal description and security analysis
  - scriptless scripts based on ECDSA

41

Previous Work

- Common to both is 2P-ECDSA:

Fast Secure Two-Party ECDSA Signing*

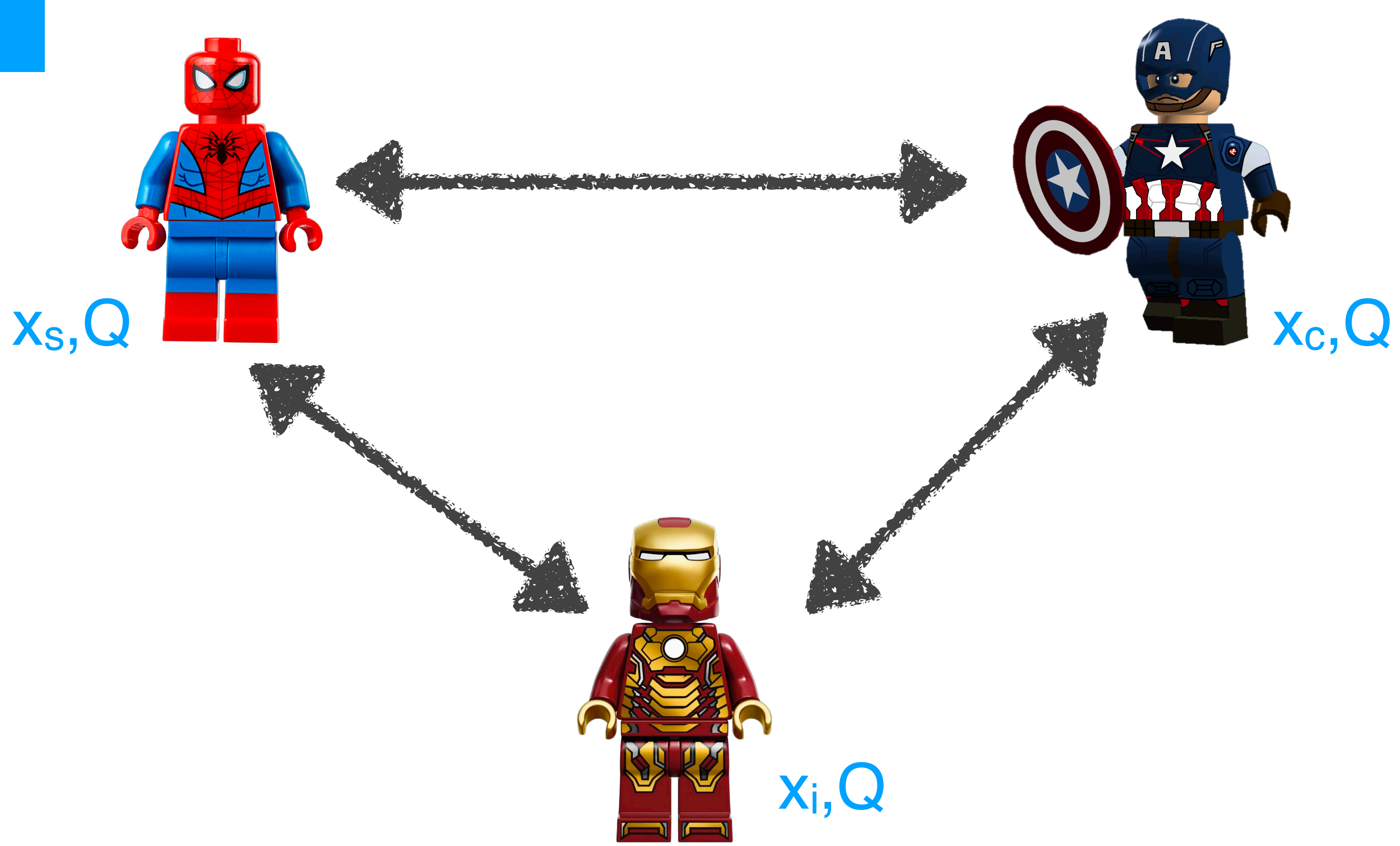
Yehuda Lindell**

Dept. of Computer Science
Bar-Ilan University, ISRAEL
`lindell@biu.ac.il`

Threshold Signatures

(t,n)-threshold signature scheme distributes signing power to n parties such that any group of at least t parties can generate a signature

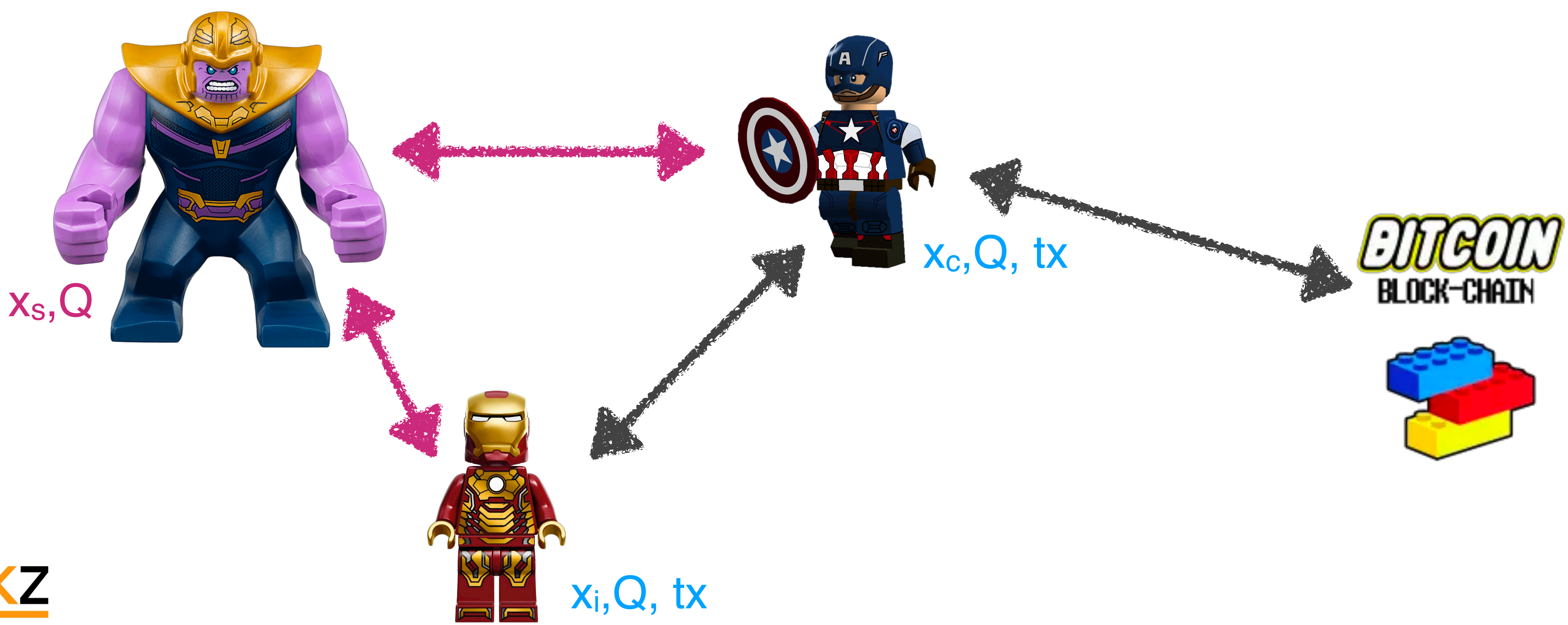
(2,3) - Keygen



Threshold Signatures

(t,n)-threshold signature scheme distributes signing power to n parties such that any group of at least t parties can generate a signature

(2,3) - Signing



2P-Keygen [L17]

Party1

$Q_1 = x_1 \cdot G$

Party2

$Q_2 = x_2 \cdot G$

Q_1 →

← Q_2

$Q = x_1 \cdot Q_2$

$Q = x_2 \cdot Q_1$

$Enc_e(x_1)$ →

The protocol promises: (1) Privacy, (2) Correctness

KZen-networks/multi-party-ecdsa

2P-Signing [L17]

Signing message m: $m' = \mathrm{Hash}(m)$

Party1

Party2

Party1		Party2
$R_1 = k_1 \cdot G$	R_1 →	$R_2 = k_2 \cdot G$
	← R_2	
$R = k_1 \cdot R$		$R = k_2 \cdot R_1$
$s = \mathrm{Dec}_d(s^*)/k_1$	← s^*	$s^* = \mathrm{Enc}_e(m' / k_2) \boxplus \mathrm{Enc}_e(x_1) \odot \mathrm{Enc}_e(x_2 \cdot r / k_2)$

Output: $\boldsymbol{\sigma} = (s, r)$, s.t. $\mathrm{Verify}(\boldsymbol{\sigma}, Q, m') = 1$

The protocol promises: Unforgeability

KZen-networks/multi-party-ecdsa

2P-ECDSA Lock [MMSKM18]

message m' = Hash(m), Adaptor (t, T)

Party1

Party2

Party1: $R_1 = k_1 \cdot T$

Party1 → Party2: R_1

Party2: $R_2 = k_2 \cdot T$

Party2 → Party1: R_2

Party1: $R = k_1 \cdot R$

Party2: $R = k_2 \cdot R_1$

Party2: $s^* =$

$Enc_e(m' / k_2) \boxplus$

$Enc_e(x_1) \odot Enc_e(x_2 \cdot r / k_2)$

Party2 → Party1: s^*

Party1: $s' = Dec_d(s^*)/k_1$

Output: $\sigma' = (s' = s \cdot t, r)$, s.t. Verify$(s' \cdot t^{-1}, r, Q, m') = 1$

KZen-networks/multi-hop-locks

Possible Issues

- Given the non-linearity of ECDSA, Are ECDSA Scriptless Scripts
  - Possible ? YES, using Lindell 2P-ECDSA
  - Possible but Inefficient ? MAYBE
  - Possible but with compromise on Security ? YES

Possible Issues #2

Digital signing by utilizing multiple distinct signing keys, distributed between two parties

Abstract

Described herein is a method and system for digital signing by utilizing Elliptic Curve Digital Signature Algorithm (ECDSA) with a group generator of an elliptic-curve group of order and an elliptic curve point Q. The method may be configured to receive a digital message and associated with a request from a third-party in order to sign the digital message. The system designed to sign such messages may comprise two parties denoted P1 and P2 configured to conduct a multiparty signing procedure by utilizing ECDSA. The digital signing procedure may follow preliminary steps configured to set the system with the necessary conditions for the multiparty signing procedure. Such preliminary steps may set the parties P1, and P2, in accordance with the configuration defined herein.

US20180359097A1

United States

Download PDF Find Prior Art

Inventor: Yehuda LINDELL

Current Assignee : Bar Ilan University

Bad News?

Threshold ECDSA Papers Circa 2017-Today

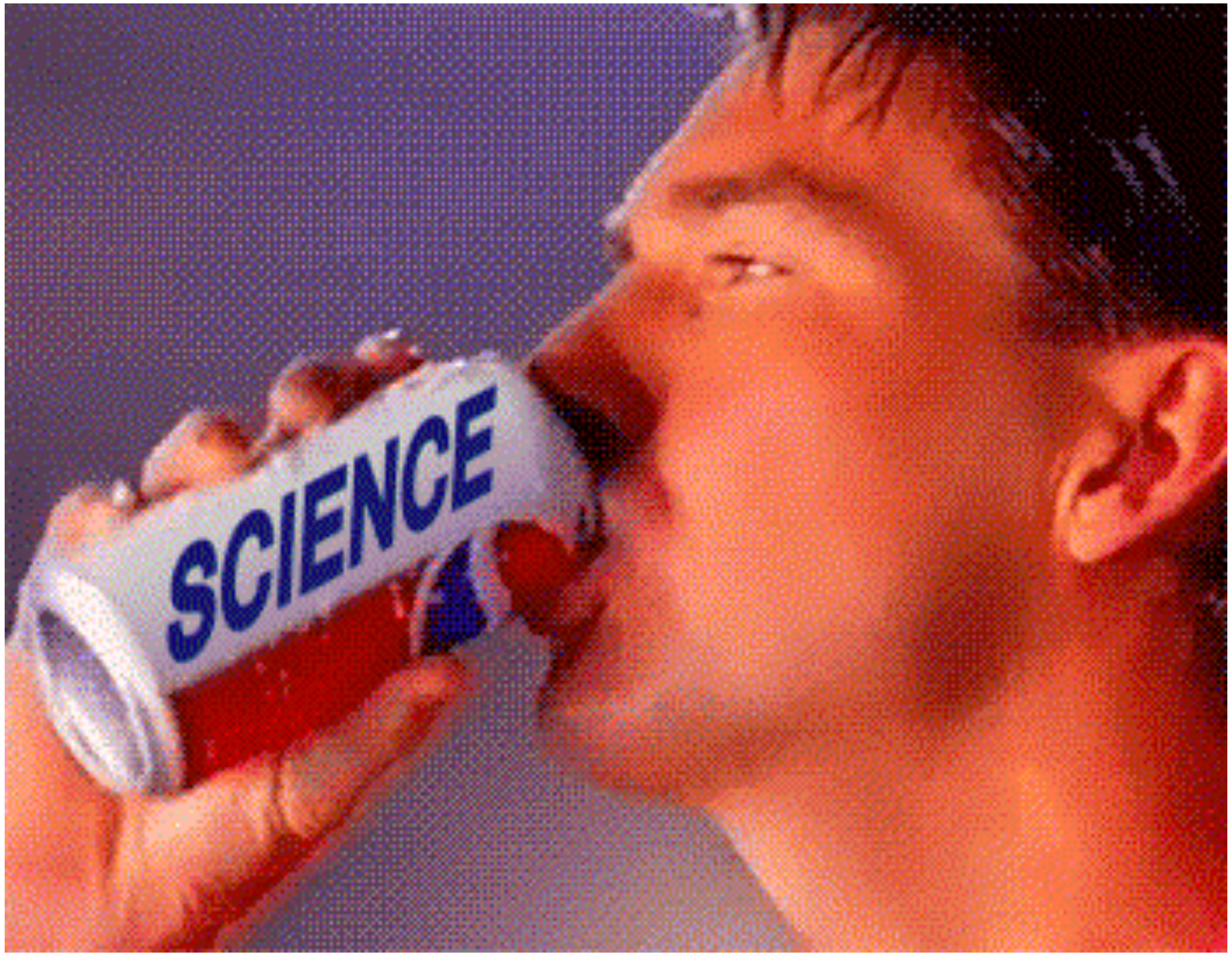
	Params	Assumptions	Signing Rounds	Signing Time
[L17]	2/2	ECDSA, Paillier	4	milliseconds
[GG18]	t/n	ECDSA, Strong RSA	9	milliseconds
[LNR18]	t/n	ECDSA, DDH	8	milliseconds
[DKLS18]	2/n	ECDSA	2	milliseconds
[DKLS19]	t/n	ECDSA	Log(t) + 6	milliseconds
[CCLST19]	2/2	ECDSA, Class groups	4	milliseconds
[SA19]	n/n	ECDSA	1	
[DKOSS19]	t/n	ECDSA	1	sub millisecond

Threshold ECDSA Papers Circa 2017-Today

	Params	Assumptions	Signing Rounds	Signing Time
[L17]	2/2	ECDSA, Paillier	4	milliseconds
[GG18]	t/n	ECDSA, Strong RSA	9	milliseconds
[LNR18]	t/n	ECDSA, DDH	8	milliseconds
[DKLS18]	2/n	ECDSA	2	milliseconds
[DKLS19]	t/n	ECDSA	Log(t) + 6	milliseconds
[CCLST19]	2/2	ECDSA, Class groups	4	milliseconds
[SA19]	n/n	ECDSA	1	
[DKOSS19]	t/n	ECDSA	1	sub millisecond

KZ

Experiments



Use Case #1: Scriptless Script MultiSig

- Access policy privacy
- Cost: one standard transaction
- Max number of parties

KZen-networks/multi-party-ecdsa

Use Case #2: Threshold Wallet

- Distributed key generation (DKG)
- Distributed Signing
- Secret Share Recovery
- Deterministic Child Address Derivation
- Rotation

[KZen-networks/gotham-city](https://github.com/KZen-networks/gotham-city)

[unbound-tech/blockchain-crypto-mpc](https://github.com/unbound-tech/blockchain-crypto-mpc)

Use Case #3: Coin Join Mixer

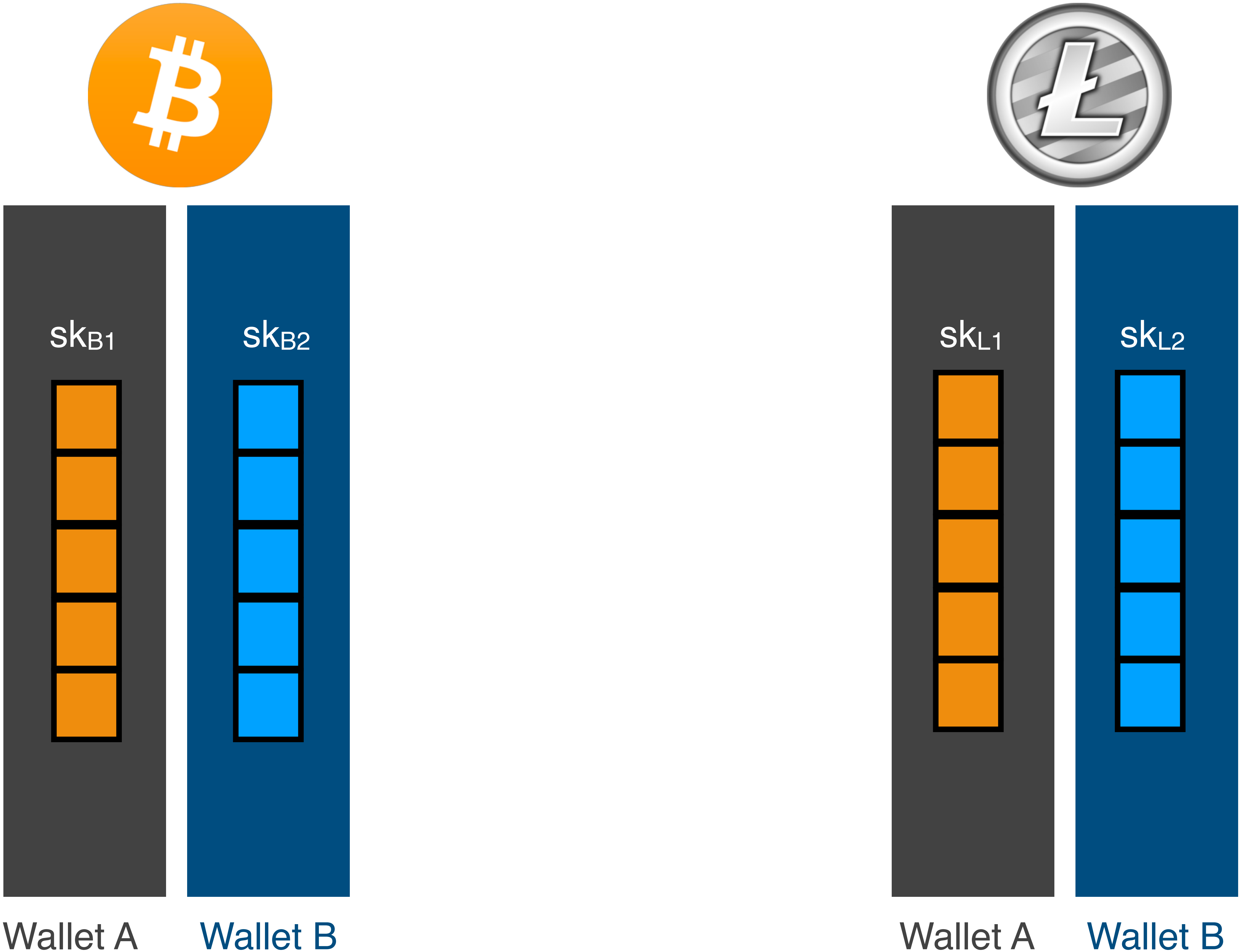
- Pubkey is equal to a sum of locally generated public keys:
  - $Pk = Pk_1 + Pk_2 + \ldots + Pk_n$
  - Basically a constructed way for parties to reach off-chain consensus on output addresses like Chaumian coin-join
- No need for central coordinator

Use Case #4: Atomic Swaps

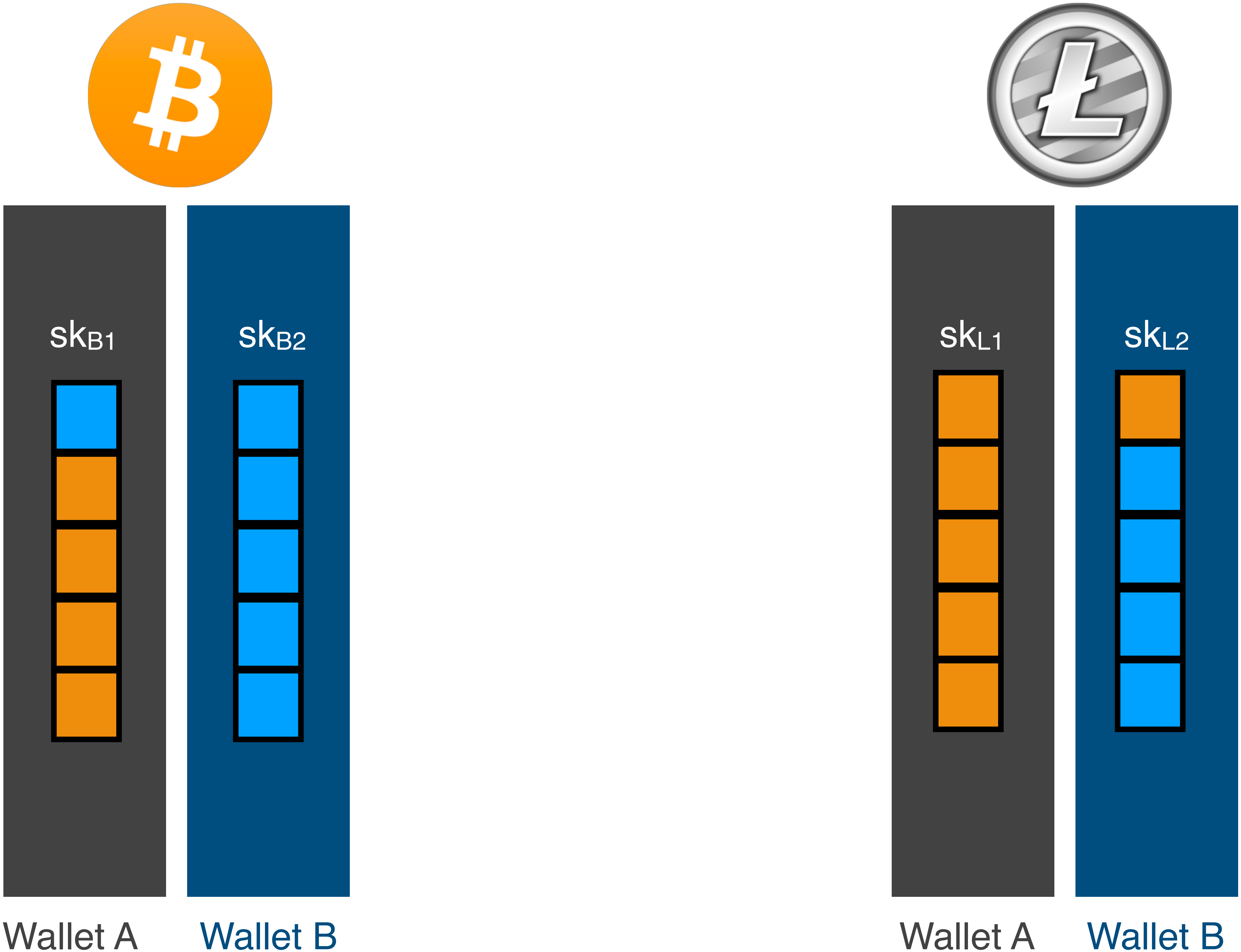
- Option 1: Use the scriptless script construction
  - Locking using Adaptor signatures.
- Option 2: Depends on access structure secret shares can be swapped using “gradual release of secrets”

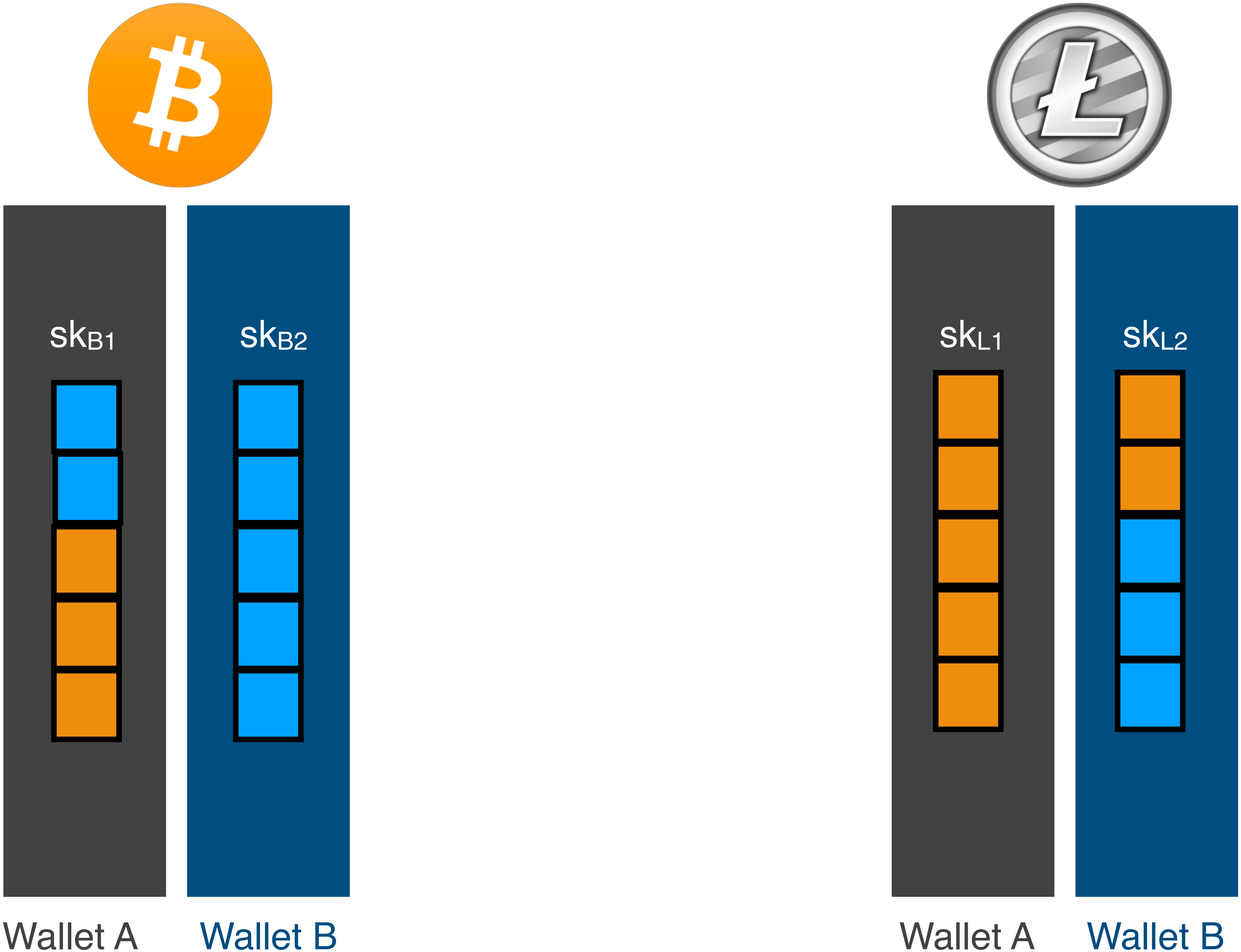
scriptless-scripts/atomic-swap

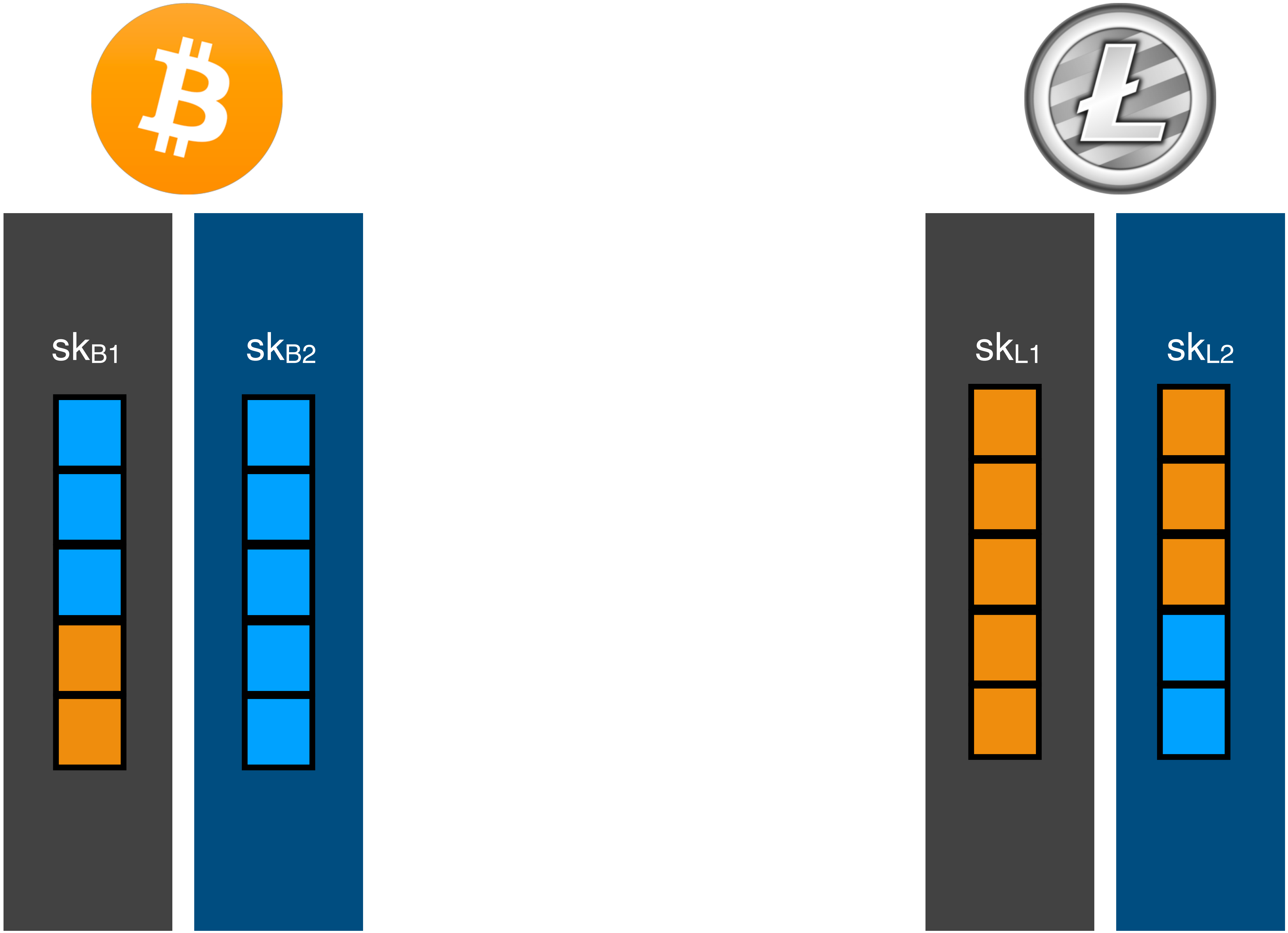
KZen-networks/centipede/

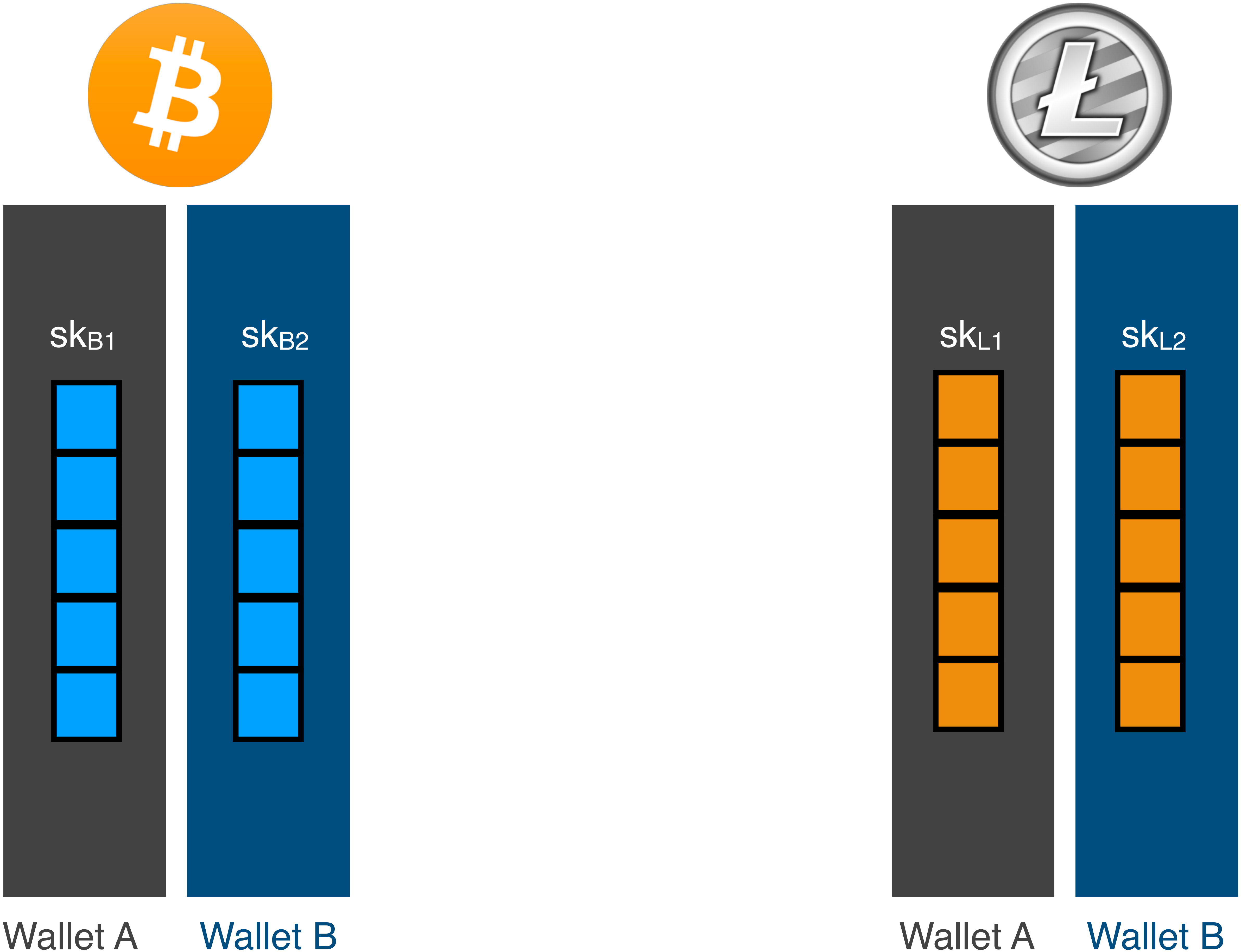


sk_{B1}
sk_{B2}
sk_{L1}
sk_{L2}
Wallet A
Wallet B
Wallet A
Wallet B


skB1
skB2
skL1
skL2
Wallet A
Wallet B
Wallet A
Wallet B


sk_{B1}
sk_{B2}
Wallet A
Wallet B
sk_{L1}
sk_{L2}
Wallet A
Wallet B


sk_{B1}
sk_{B2}
sk_{L1}
sk_{L2}

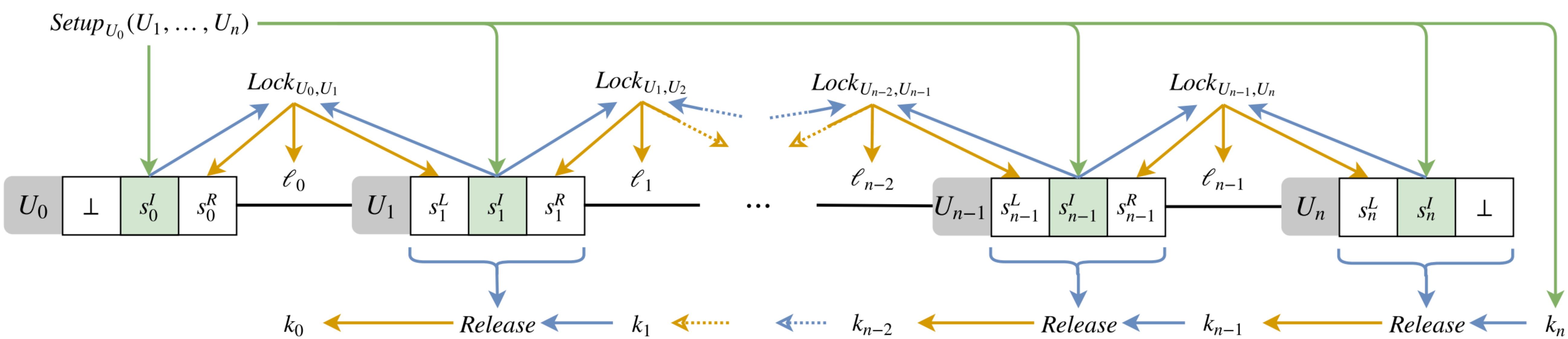


sk_{B1}
sk_{B2}
sk_{L1}
sk_{L2}
Wallet A
Wallet B
Wallet A
Wallet B

Use Case #6: Payment Channel Network

Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability

Giulio Malavolta[*§], Pedro Moreno-Sanchez[*¶†], Clara Schneidewind[†], Aniket Kate[‡], Matteo Maffei[†]
[§]Friedrich-Alexander-University Erlangen-Nürnberg, [†]TU Wien, [‡] Purdue University



KZen-networks/multi-hop-locks/

Use Case #7: Zero Knowledge Contingent Payments

Efficient Zero-Knowledge Contingent Payments in Cryptocurrencies Without Scripts

Wacław Banasik, Stefan Dziembowski, and Daniel Malinowski

University of Warsaw

ZKCP

Prover / Seller

Verifier / Buyer

Knows x

Run threshold KeyGen

Generate TX_1, TX_2:

$TX_{1,in}$: Buyer

$TX_{1,out}$: threshold-Sig

$TX_{2,in}$: TX_1

$TX_{2,out}$: Seller

Sign TX_2

$\boxed{\sigma}$

ZK Prove x is encrypted by σ

Broadcast TX_1

Broadcast TX_2

Extract x

R & D

Still long way to go!

R & D

Still long way to go!

- Threshold cryptography - In standardisation process by NIST (*)

(*)https://csrc.nist.gov/Projects/Threshold-Cryptography

R & D

Still long way to go!

- Threshold cryptography - In standardisation process by NIST (*)
- Network layer : authenticated secure p2p communication, Broadcast channel
  - Idea: use a blockchain/ consensus layer for the communication

R & D

Still long way to go!

- Threshold cryptography - In standardisation process by NIST (*)
- Network layer : authenticated secure p2p communication, Broadcast channel
    - Idea: use a blockchain/ consensus layer for the communication
- Improvements: Accountability , Batch signing and verification

Summary

- Threshold ECDSA based Scriptless Scripts / systems
  - Are possible now
  - Are practical to use in real life
  - Hold strong security guarantees and the focus of active research in the cryptography community

Summary

- Threshold ECDSA based Scriptless Scripts / systems
  - Are possible now
  - Are practical to use in real life
  - Hold strong security guarantees and the focus of active research in the cryptography community

Questions?

Special thanks: Oded Leiba, Jonas Nick, Elichai Turkel, Pedro Moreno Sanchez

ZenGo/research

https://t.me/kzen_research

https://github.com/KZen-networks

KZ