TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

S&P SECURITY & PRIVACY GROUP

Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability

(https://eprint.iacr.org/2018/472.pdf)

Giulio Malavolta*, Pedro Moreno-Sanchez†, @pedrorechez

Clara Schneidewind†, Aniket Kate¶, @aniketpkate

Matteo Maffei† @matteo_maffei

*FAU Erlangen-Nürnberg †TU Wien ¶Purdue University

Motivation: Scalability Issues

- Bitcoin has a low transaction rate (~10 tx/sec)
  - Visa, in contrast, supports >10K tx/sec
- Scalability approaches:
  - On-chain (consensus layer or layer 1): e.g., Sharding
  - Off-chain (application layer or layer 2): e.g., Payment Channel Networks

Motivation: Scalability Issues

- Bitcoin has a low transaction rate (~10 tx/sec)
  - Visa, in contrast, supports >10K tx/sec
- Scalability approaches:
  - On-chain (consensus layer or layer 1):
    e.g., Sharding
  - Off-chain (application layer or layer 2):
    e.g., Payment Channel Networks

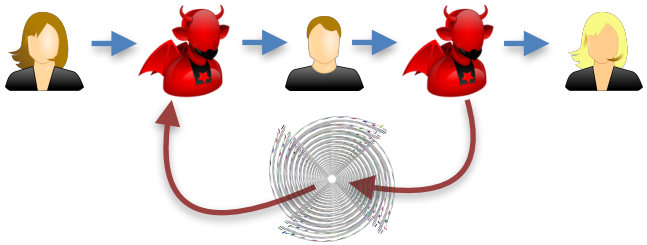
Lightning Network
(Bitcoin)

Raiden Network
(Ethereum)

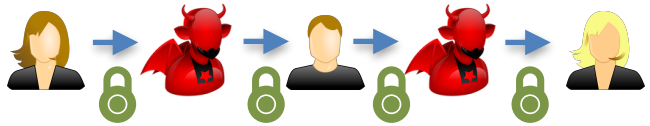
Many other research projects (Bolt, Z-Channels, Perun, etc.)

Contributions

The **Wormhole Attack**:
A novel attack on Payment Channel Network Security



AMHLs: A **new primitive** for secure + anonymous Payment Channel Networks

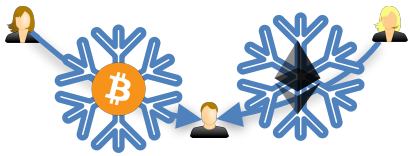


Concrete **constructions** of AMHLs that

… are efficient

… got implemented in Bitcoin's Lightning Network

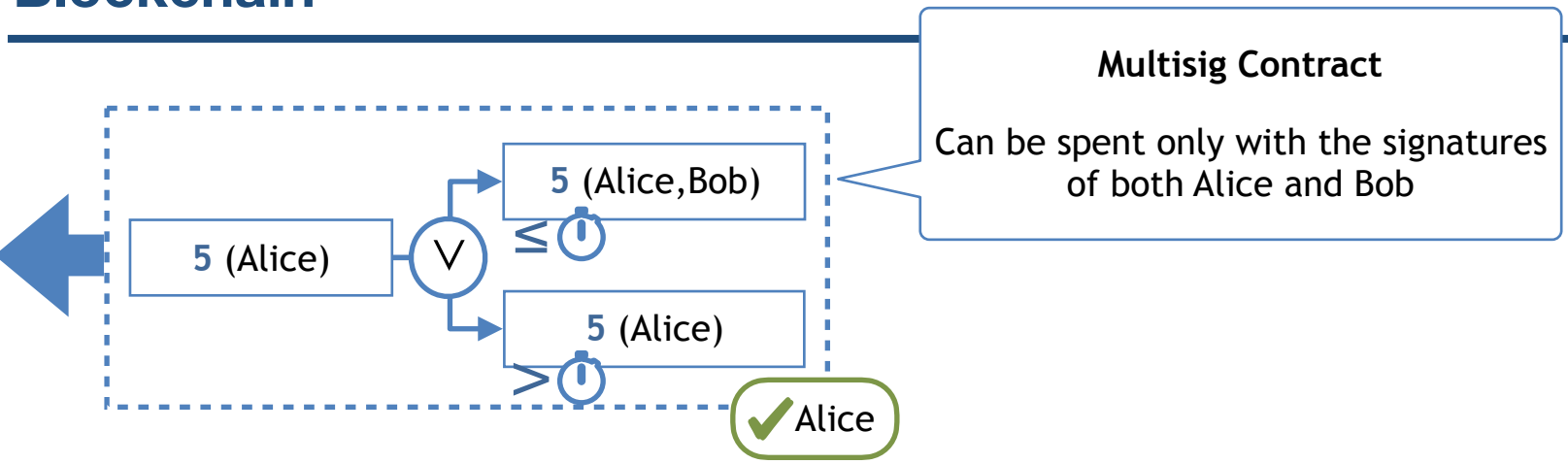
… enable inter-blockchain Payment Channels



Background on Payment Channel Networks

Payment Channels: Open

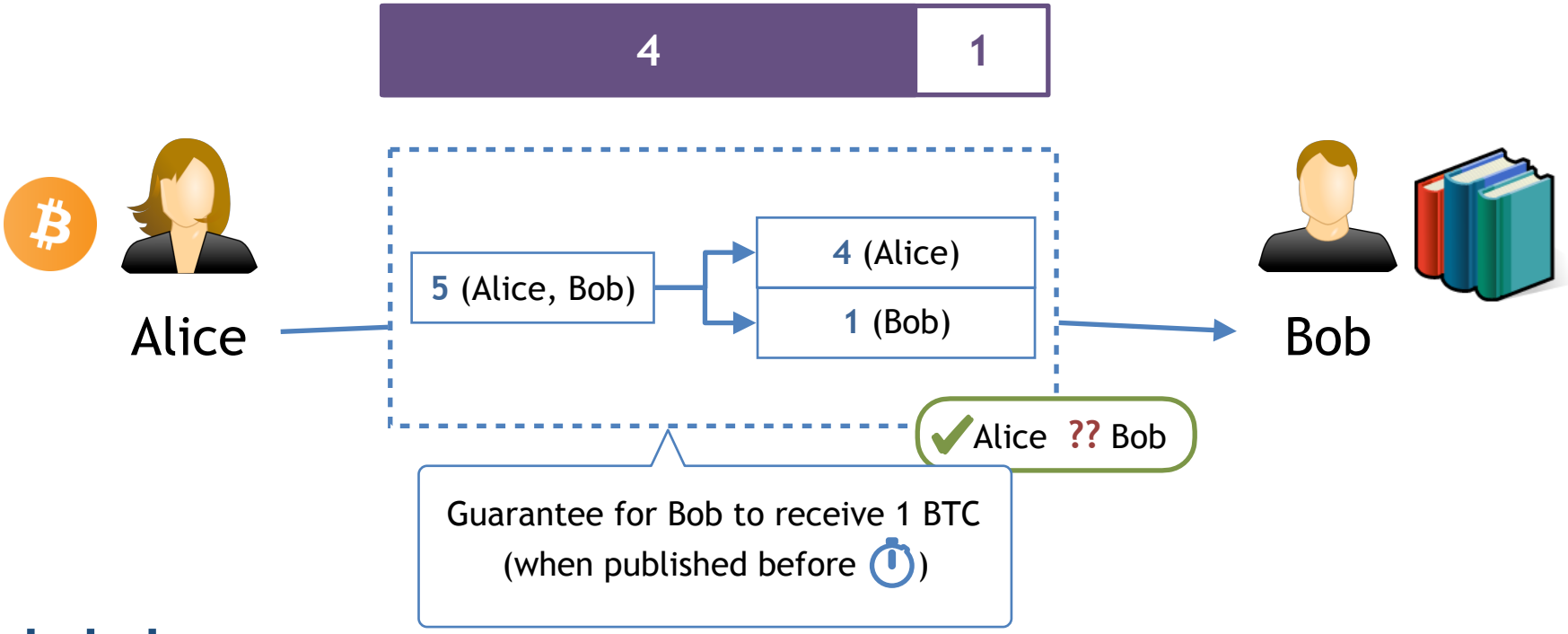
Blockchain



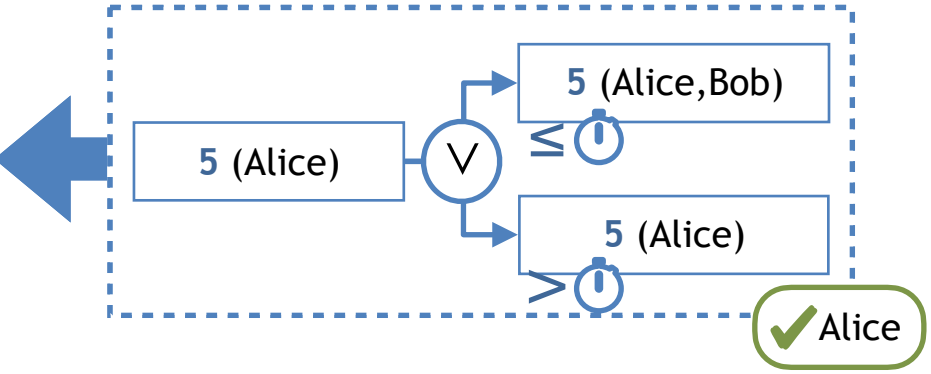
Multisig Contract

Can be spent only with the signatures of both Alice and Bob

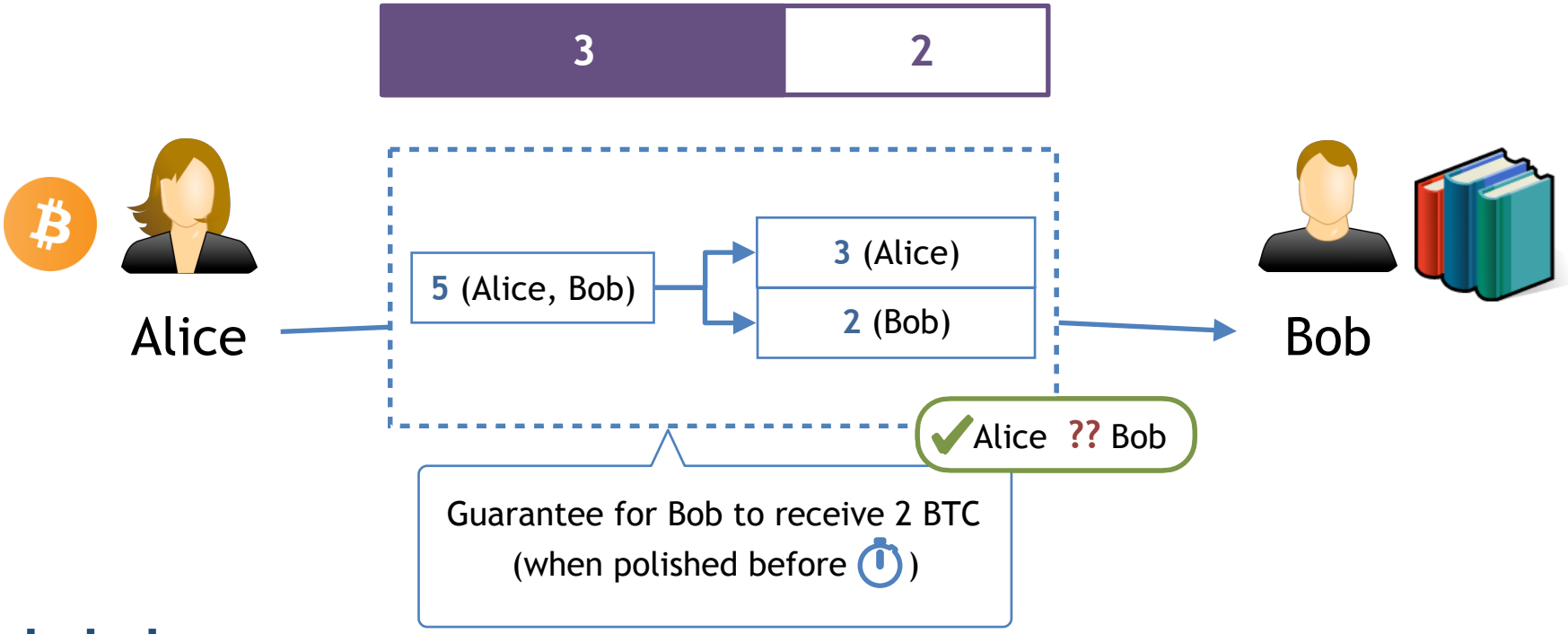
Payment Channels: Transactions



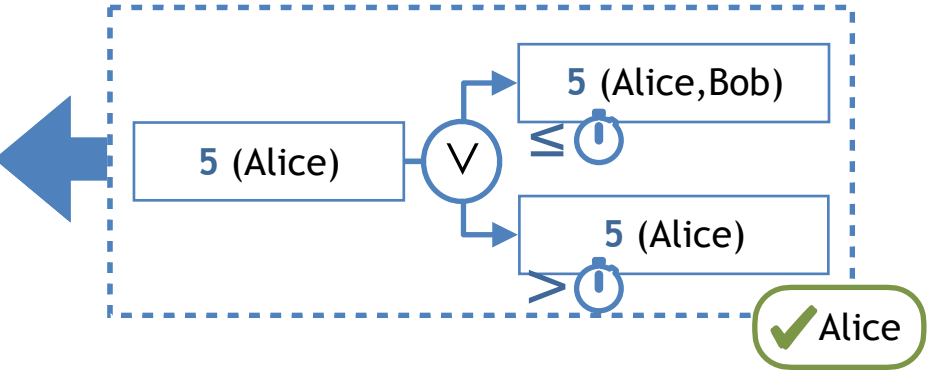
Blockchain



Payment Channels: Transactions

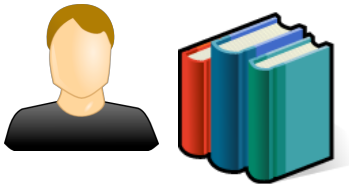


Blockchain



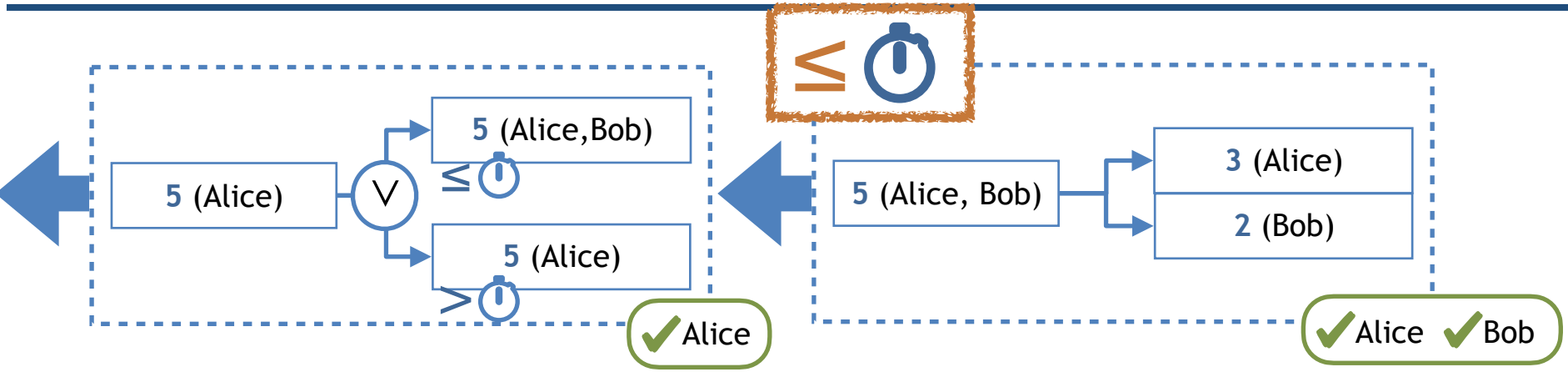
Payment Channels: Close

Alice

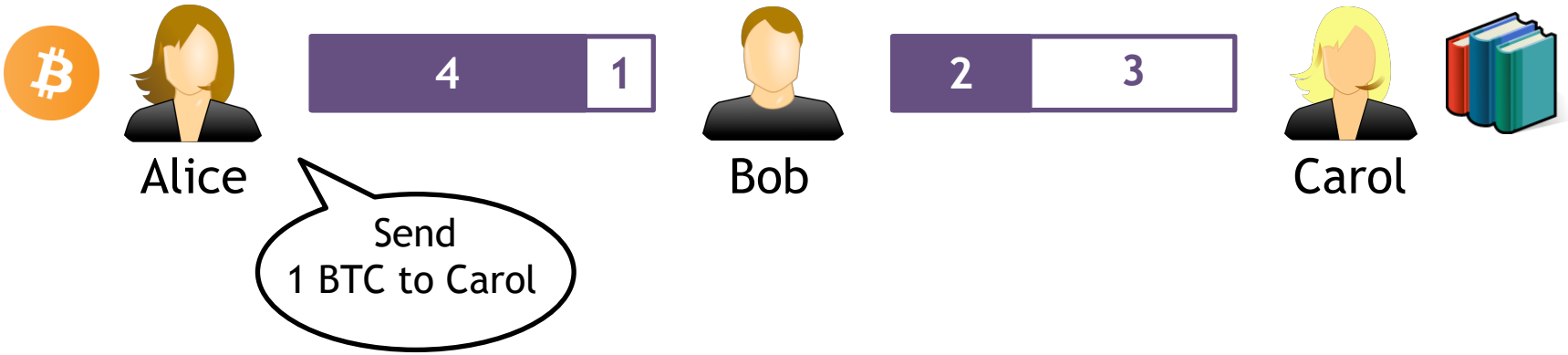


Bob

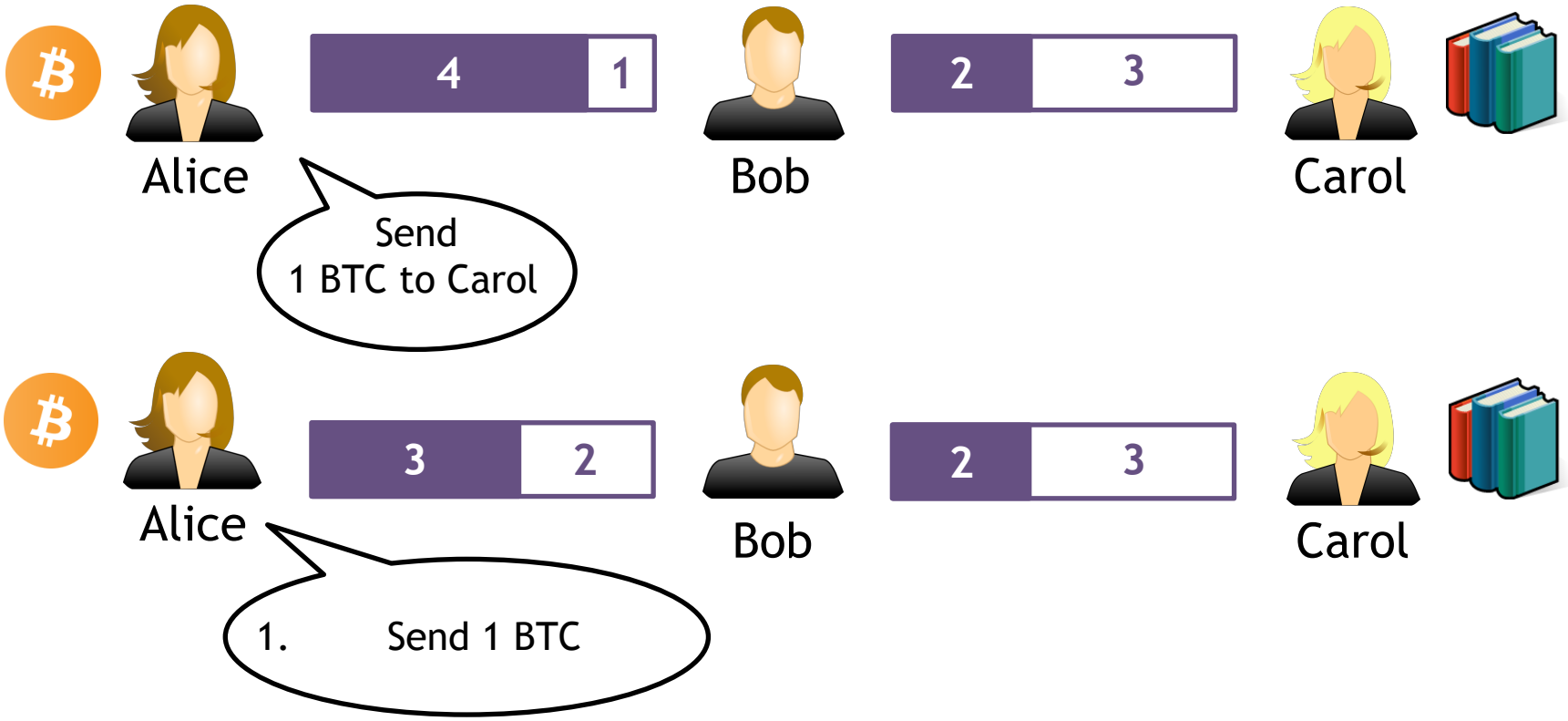
Blockchain



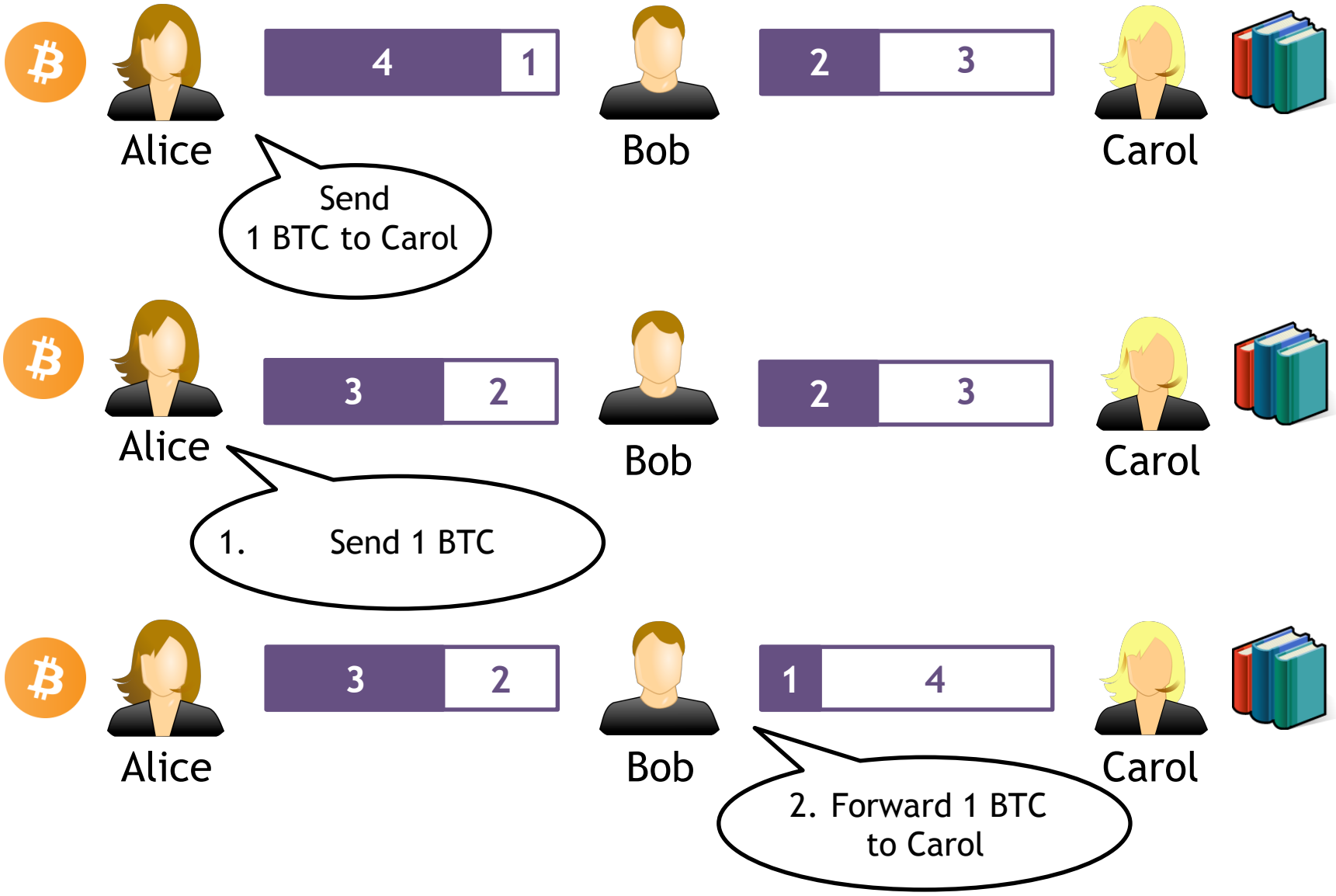
Payment Channel Networks



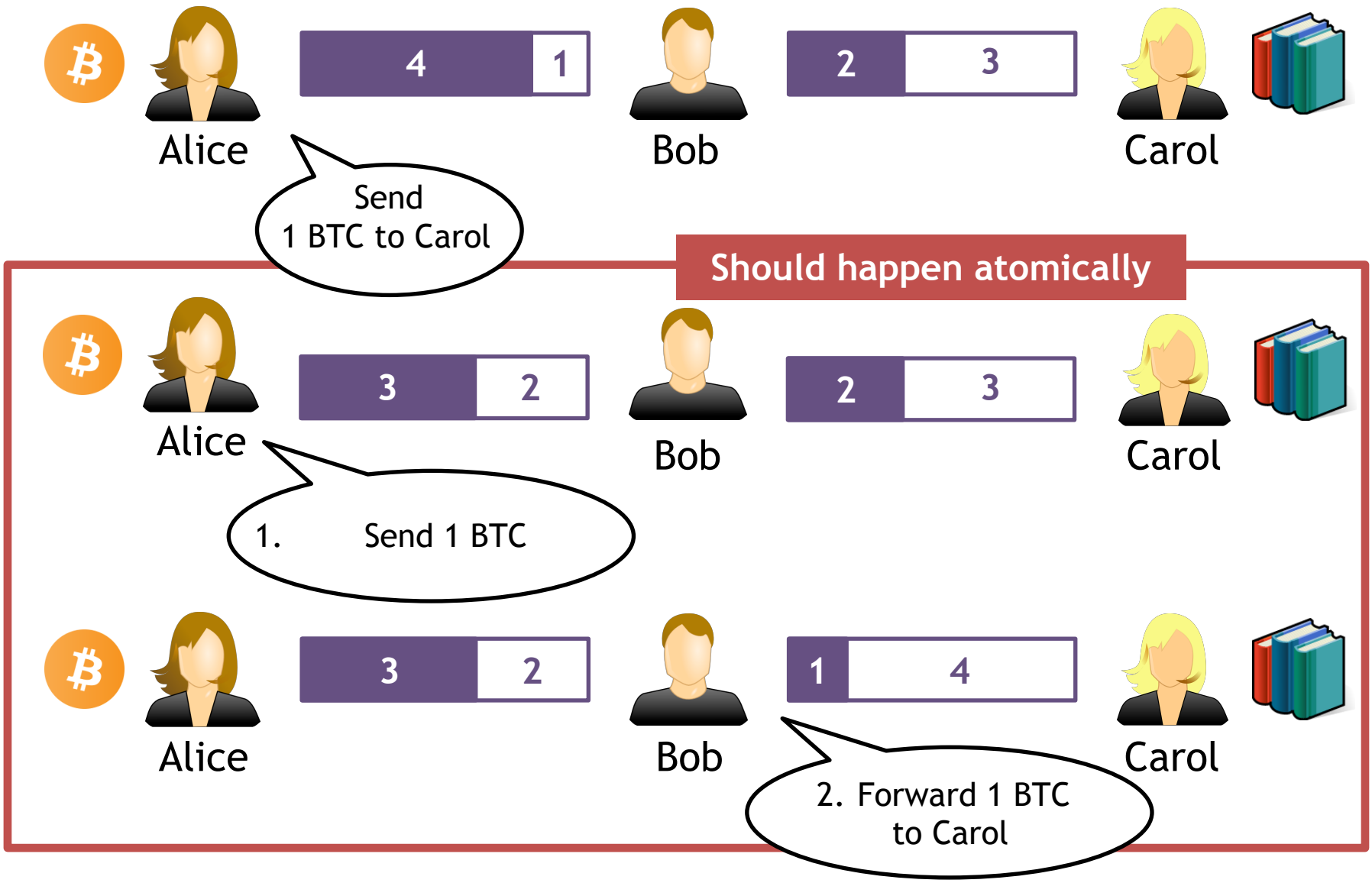
Payment Channel Networks



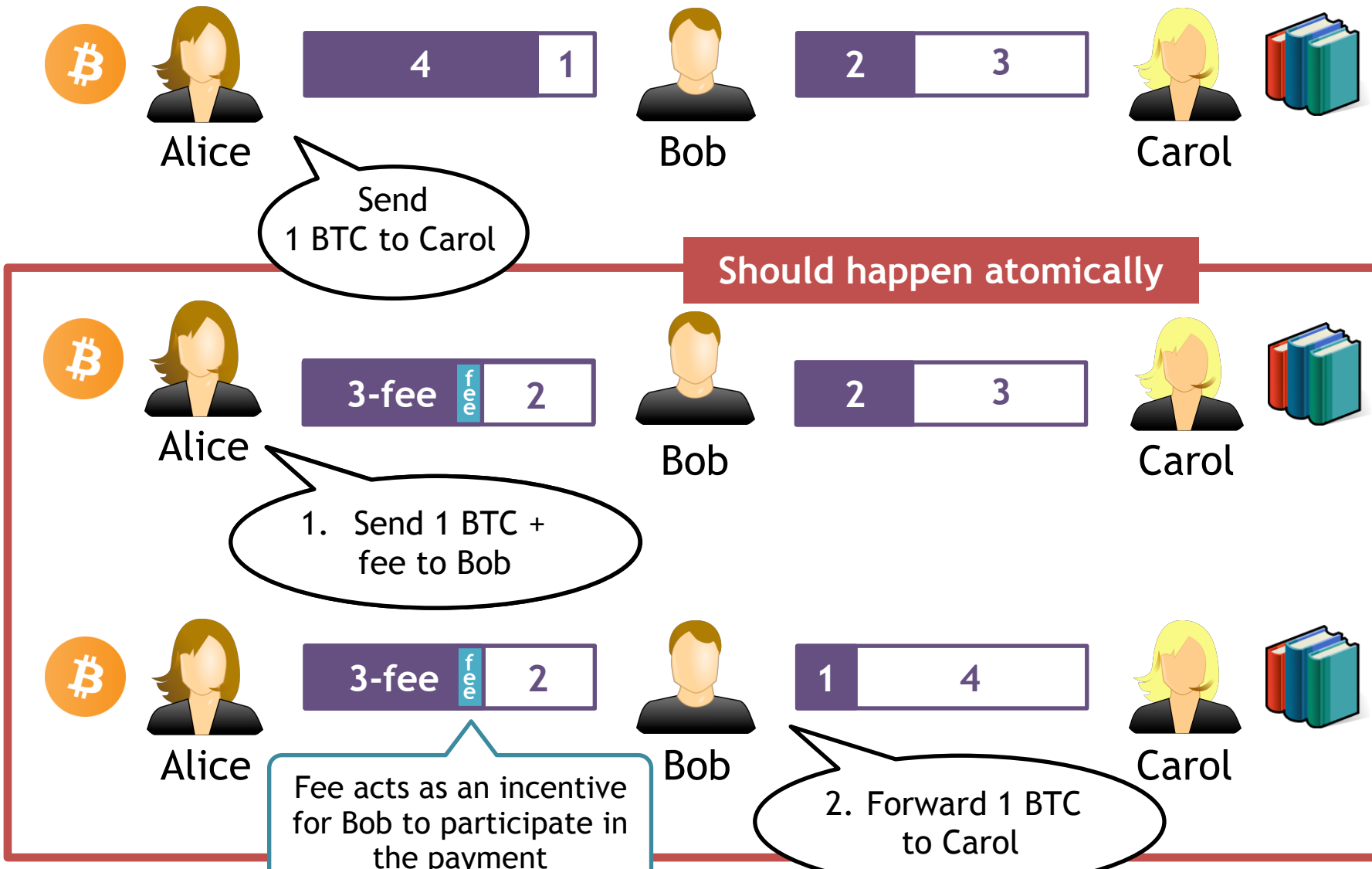
Payment Channel Networks



Payment Channel Networks

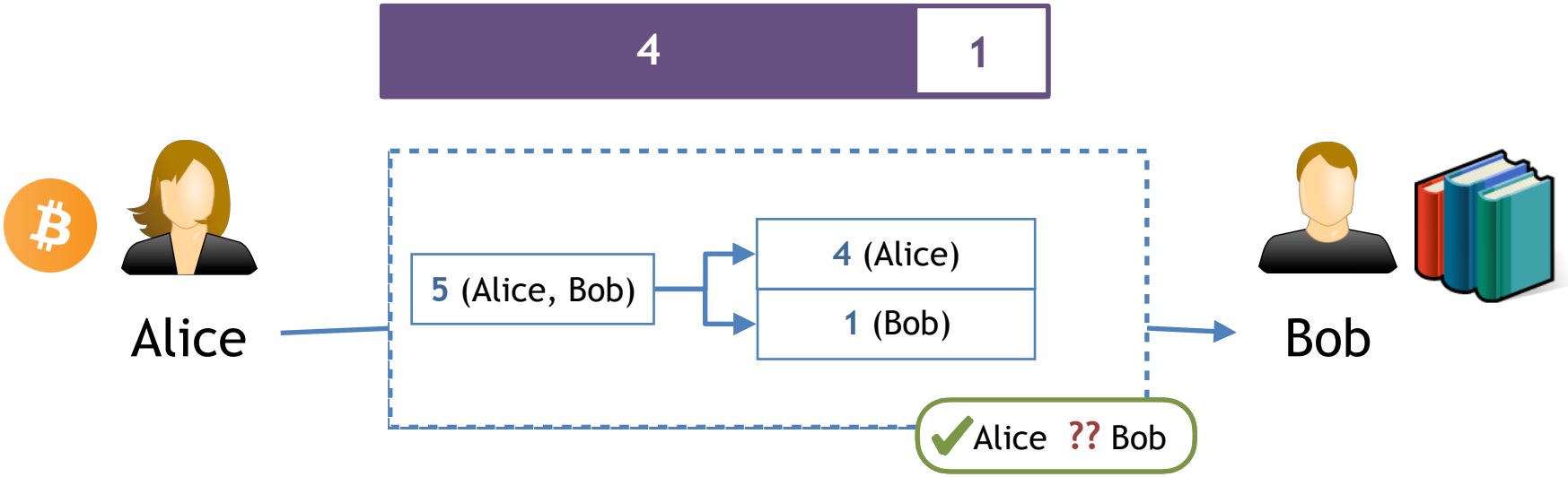


Payment Channel Networks

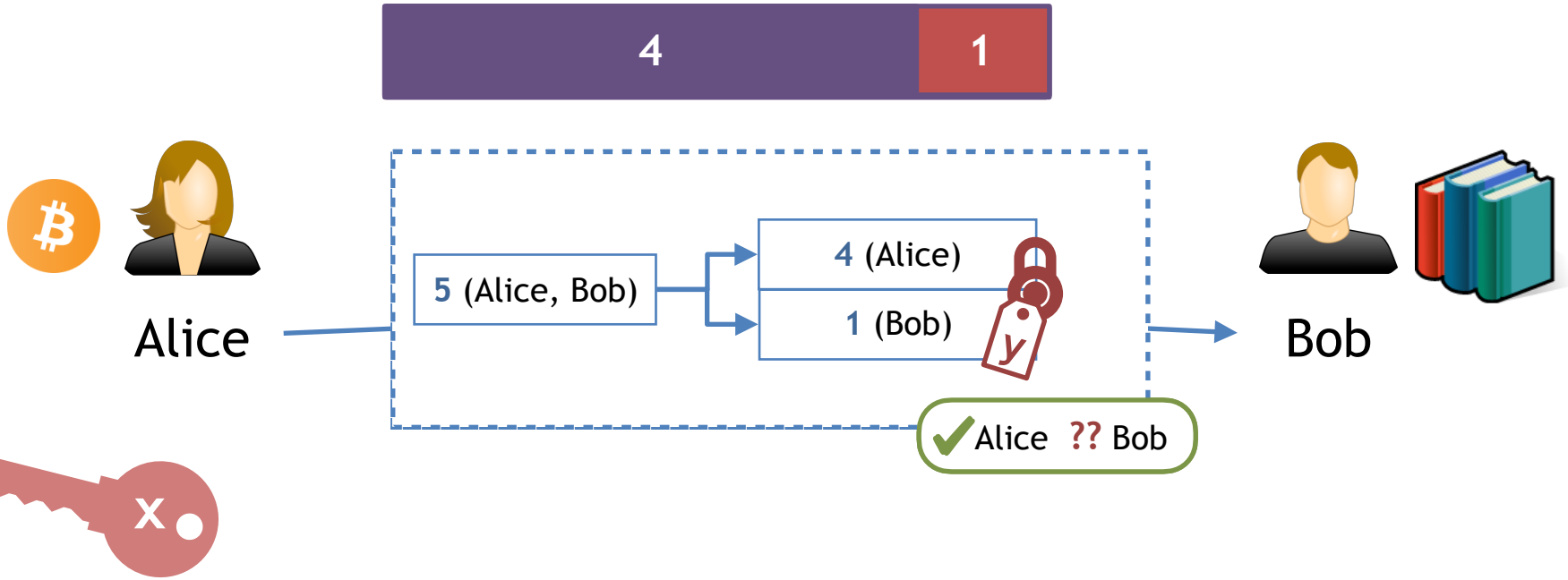


The Lightning Network (LN)

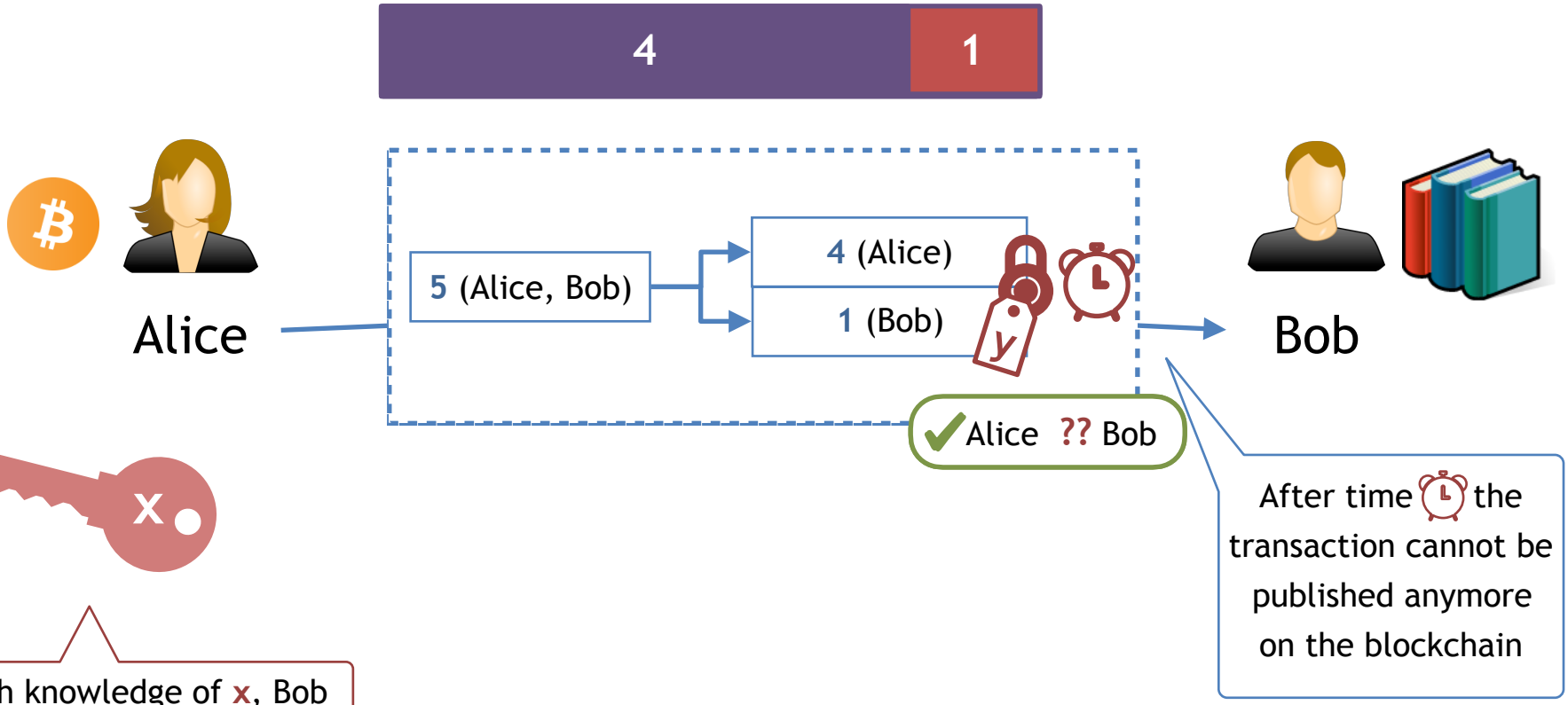
Payment Channels in the LN



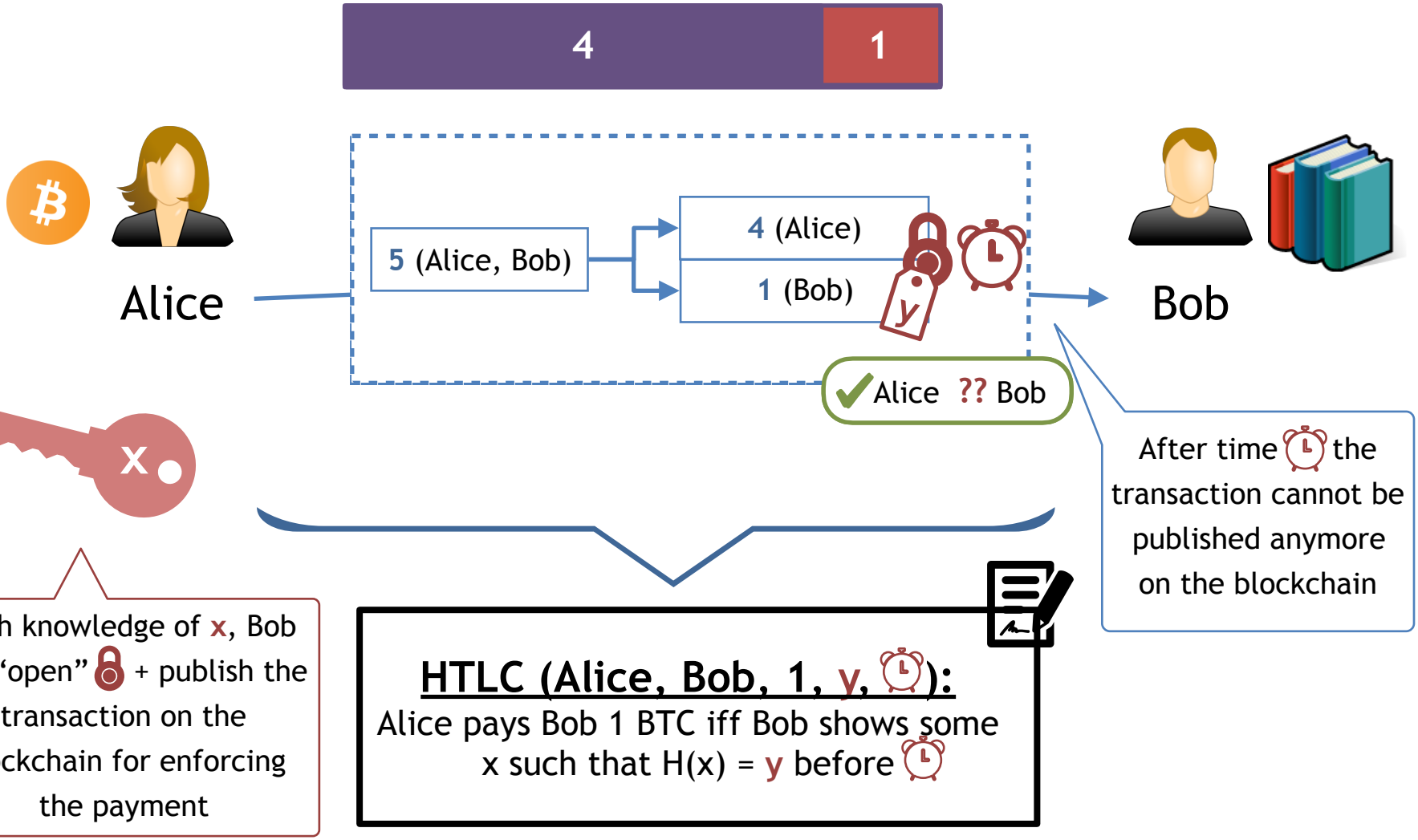
Payment Channels in the LN



Payment Channels in the LN

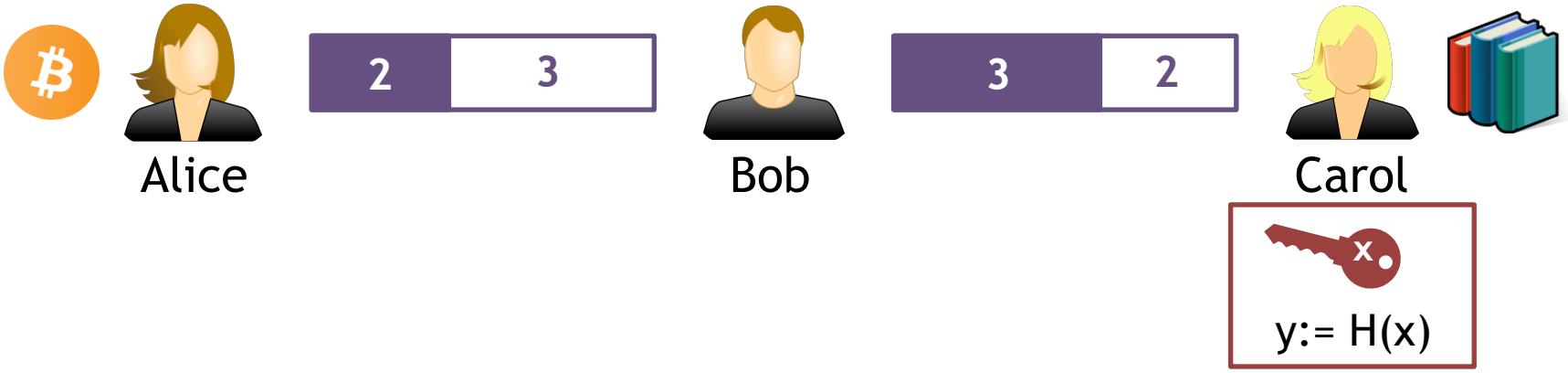


Payment Channels in the LN

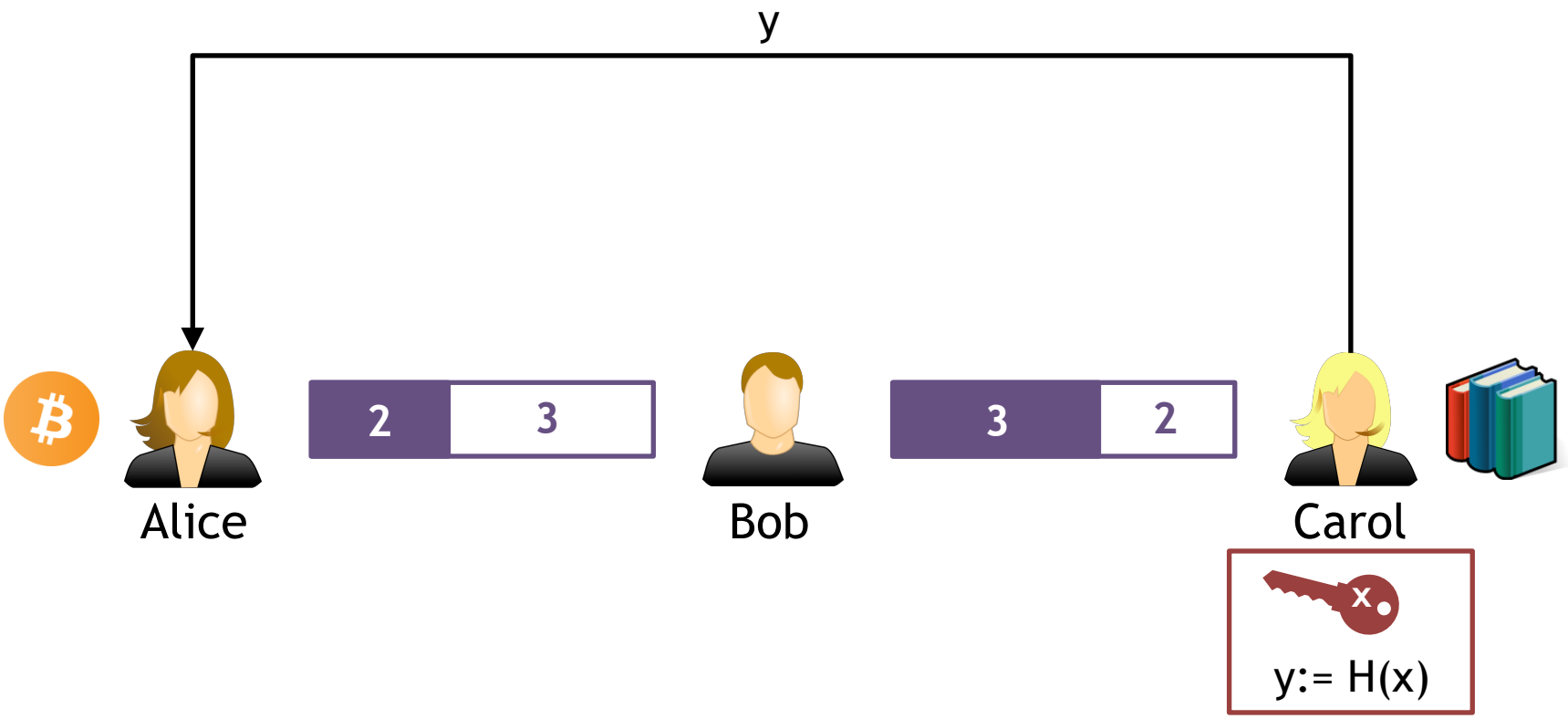


"Multi-hop-Lock"

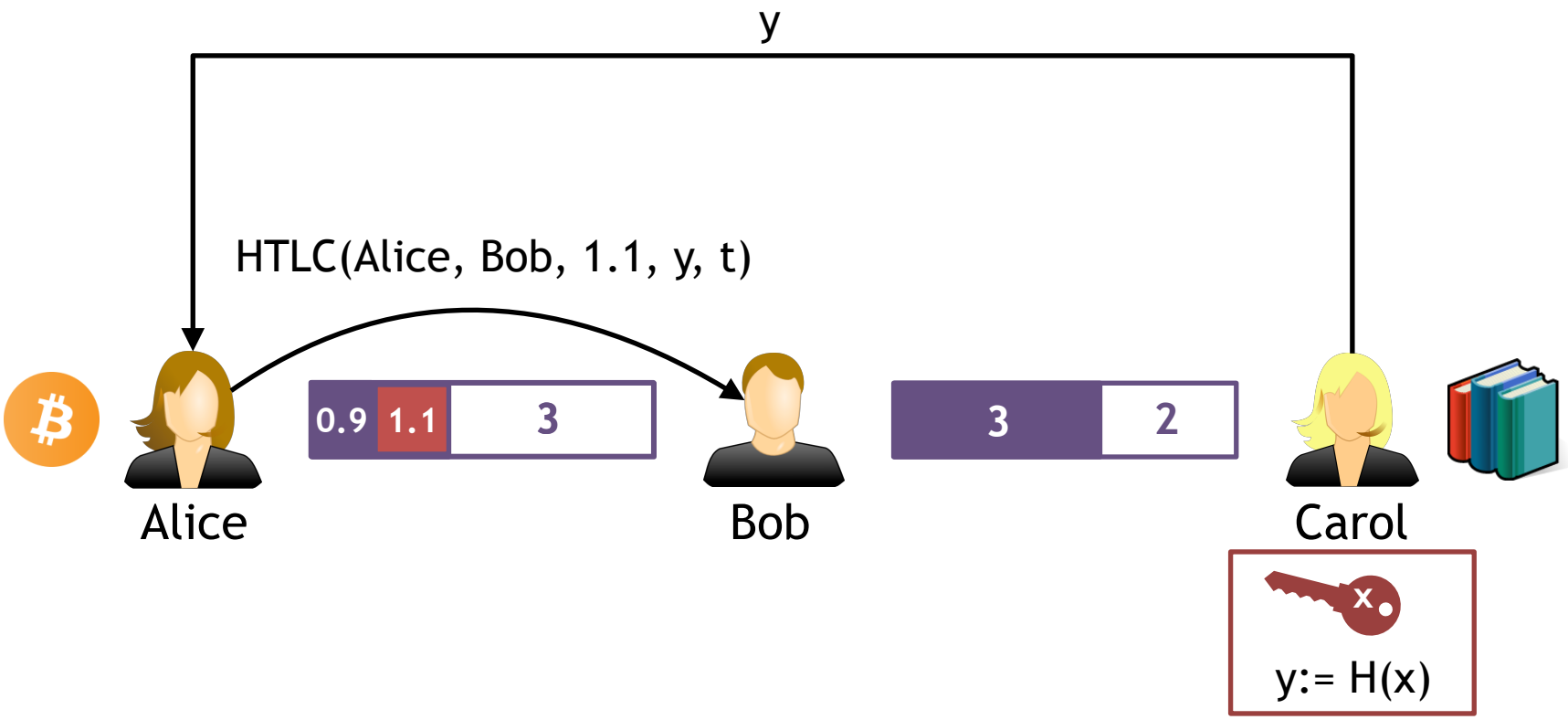
HTLC for Multi-hop Payments



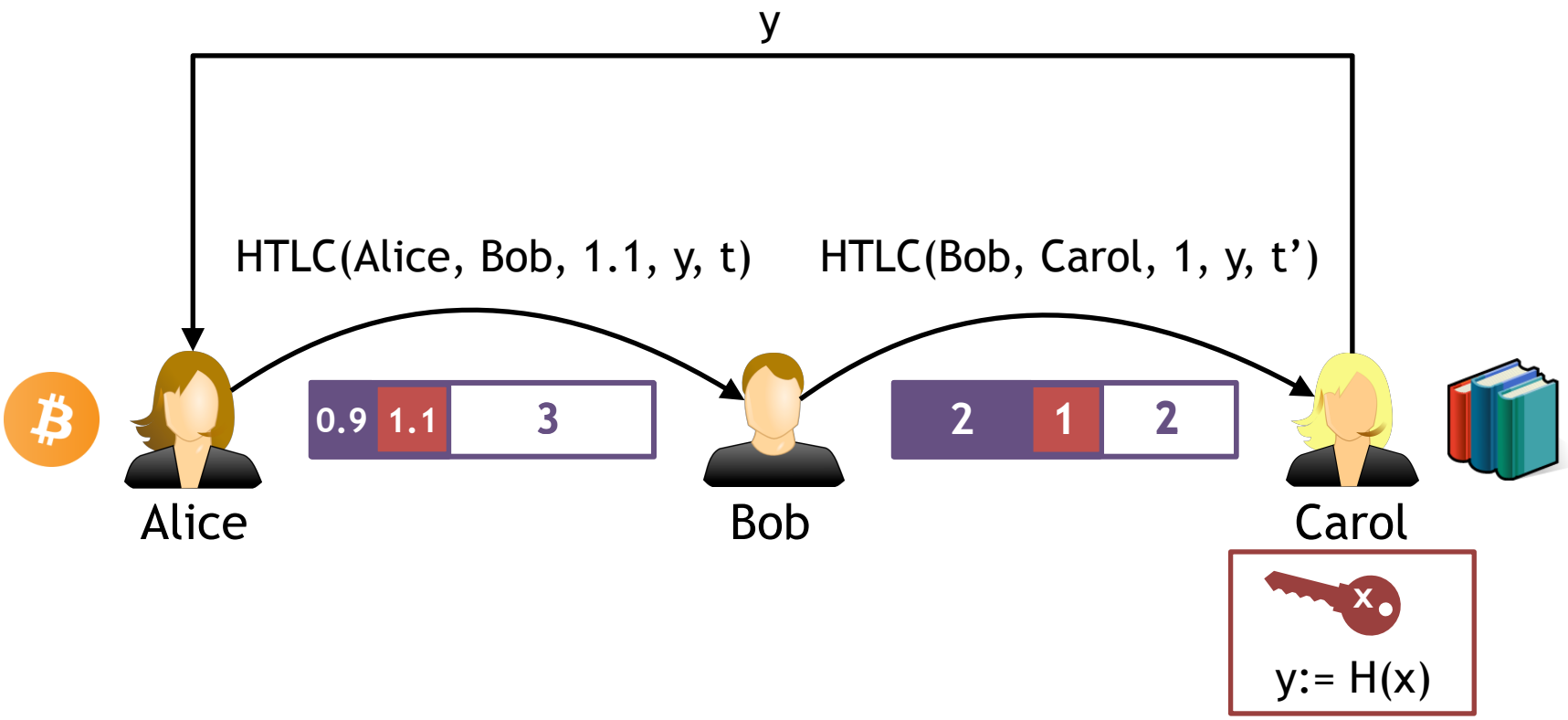
HTLC for Multi-hop Payments



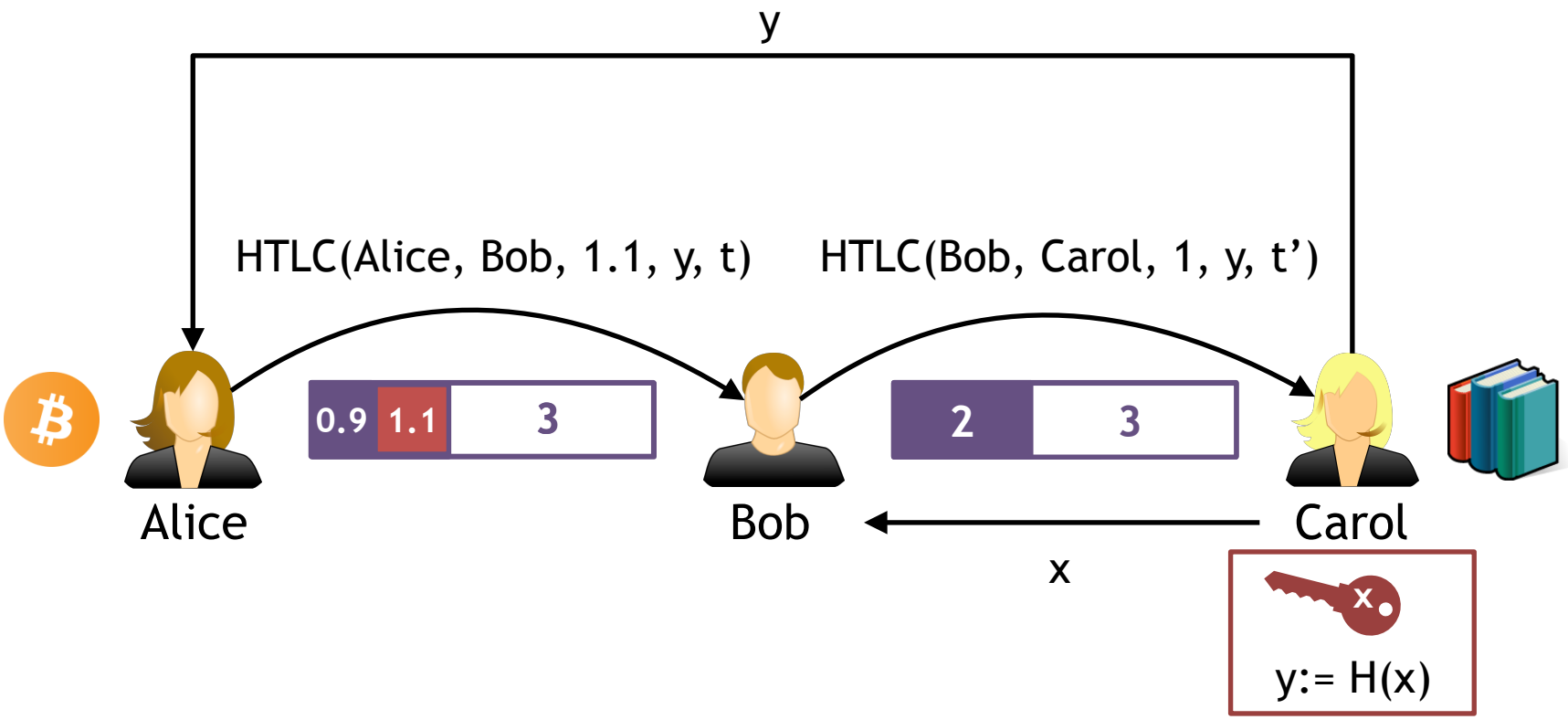
HTLC for Multi-hop Payments



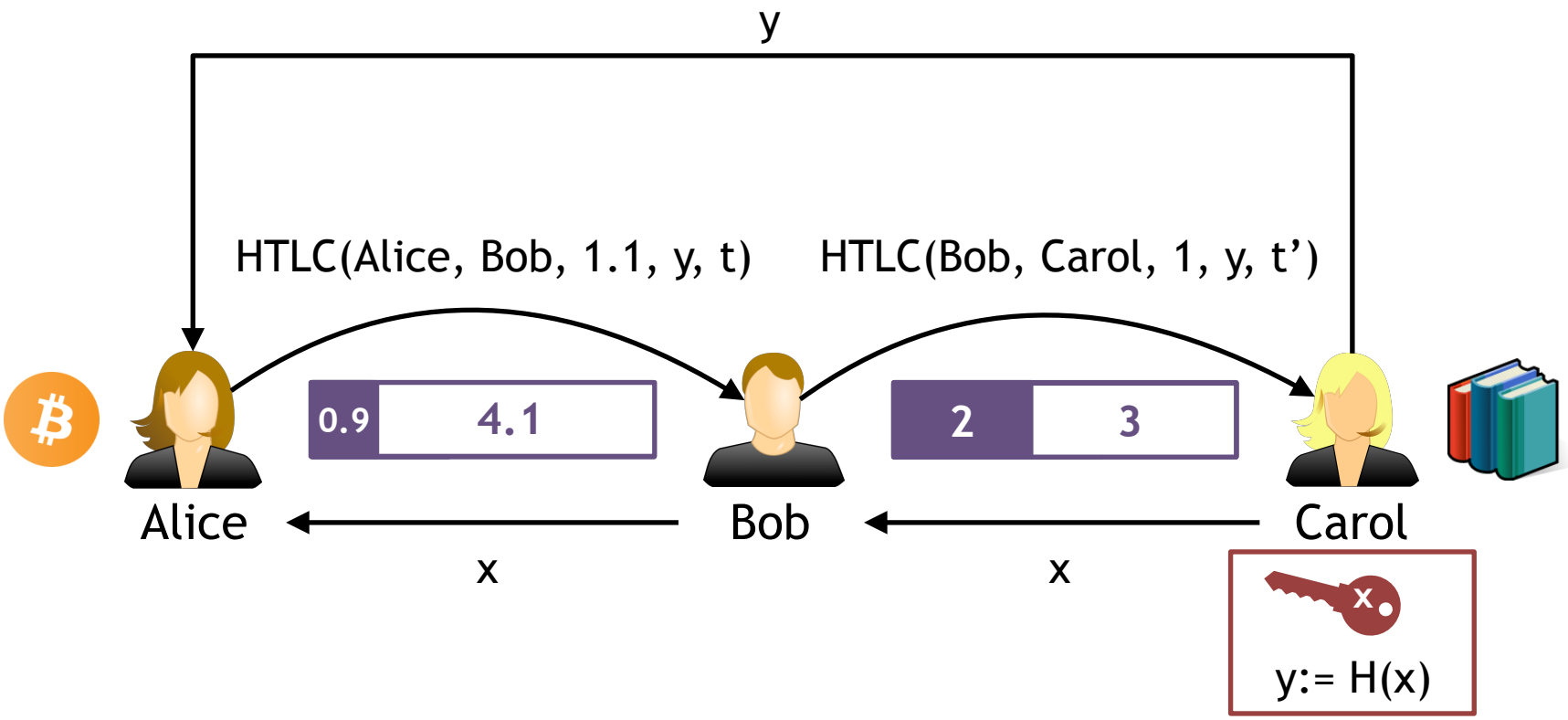
HTLC for Multi-hop Payments



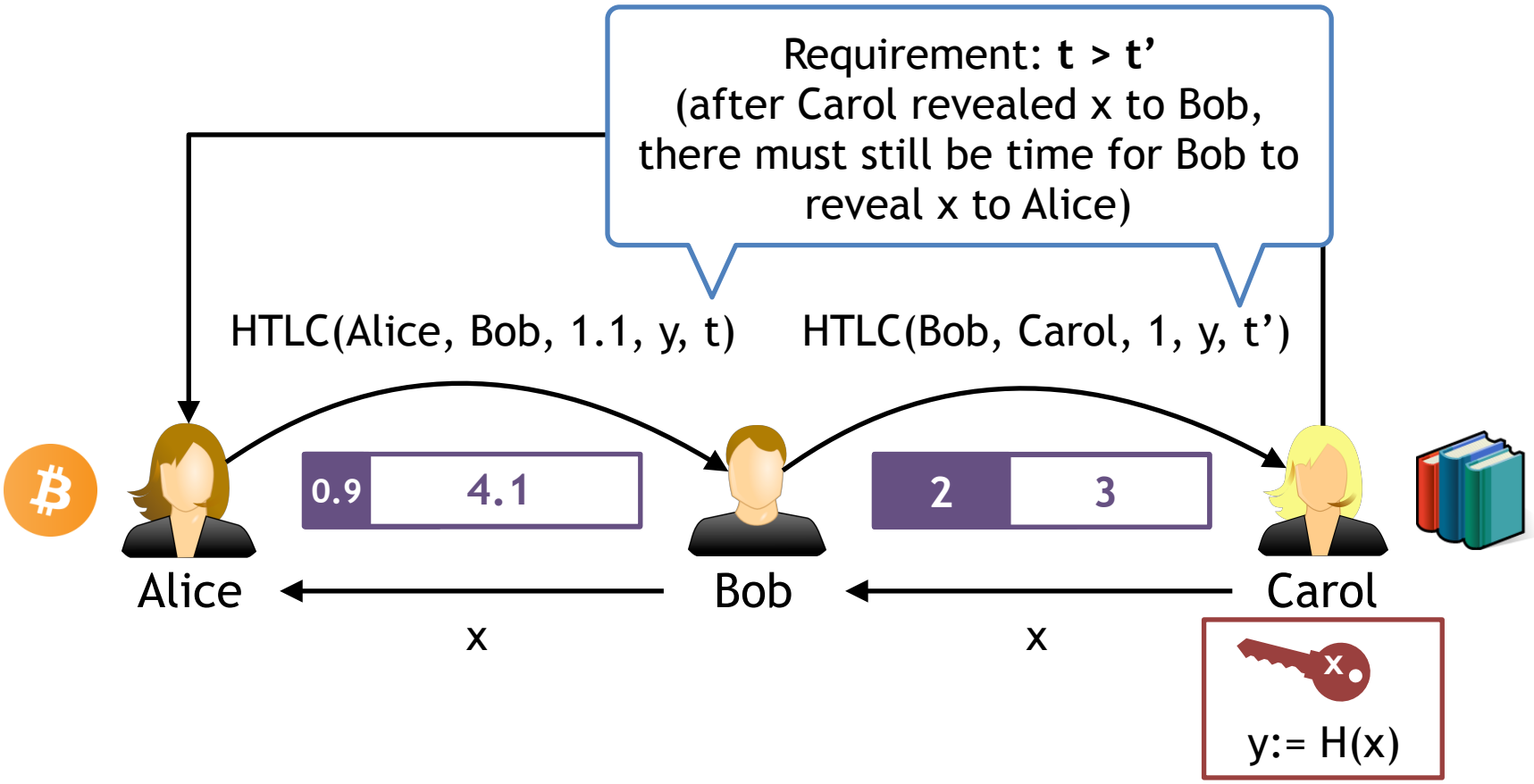
HTLC for Multi-hop Payments



HTLC for Multi-hop Payments



HTLC for Multi-hop Payments



Security and Privacy Issues in Existing PCNs

Security + Privacy in PCNs

Are off-chain payments in PCNs secure?

(No honest participant looses money!)

Are off-chain payments in PCNs privacy-preserving by default?

(individual payments are not recorded on the blockchain!)

Security + Privacy in PCNs

Are off-chain payments in PCNs secure?

(No honest participant looses money!)

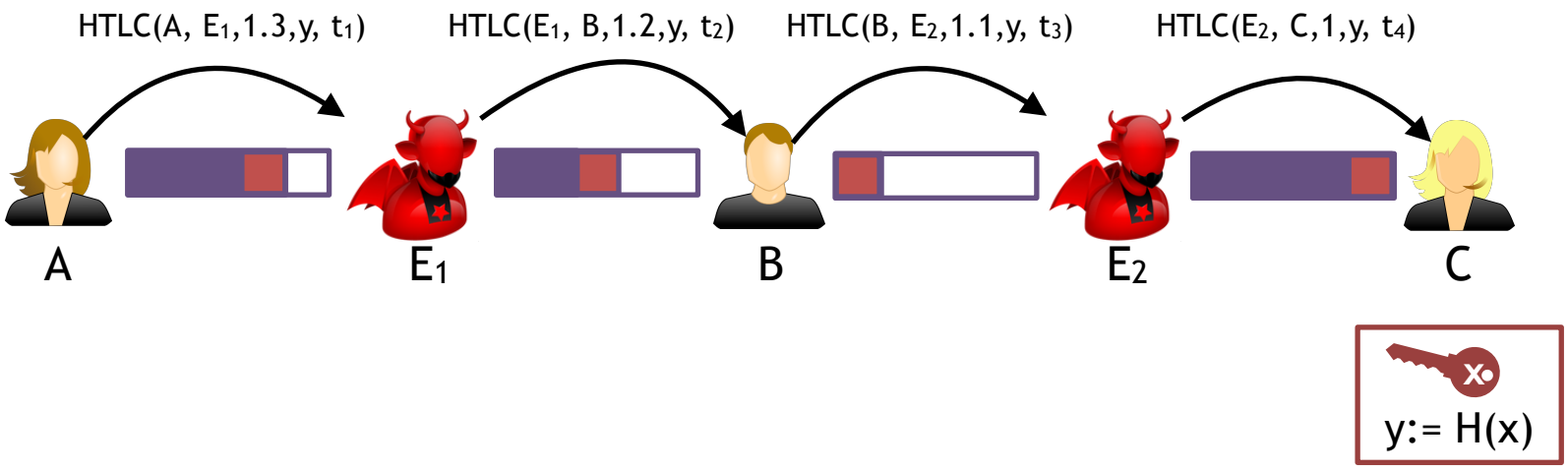
NO!

Are off-chain payments in PCNs privacy-preserving by default?

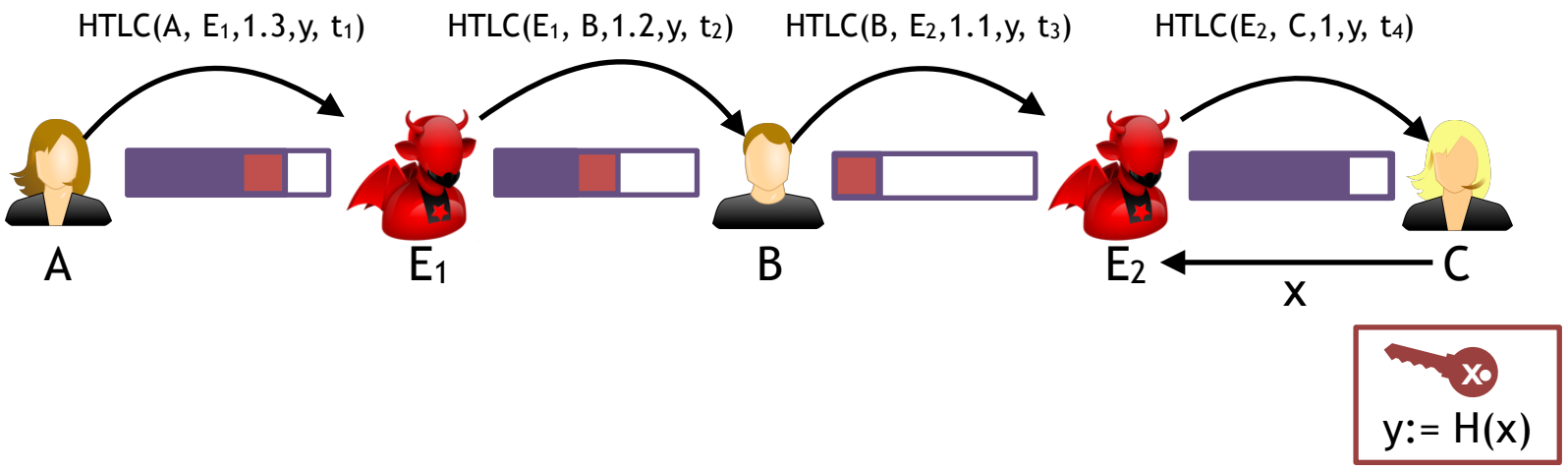
(individual payments are not recorded on the blockchain!)

NO!

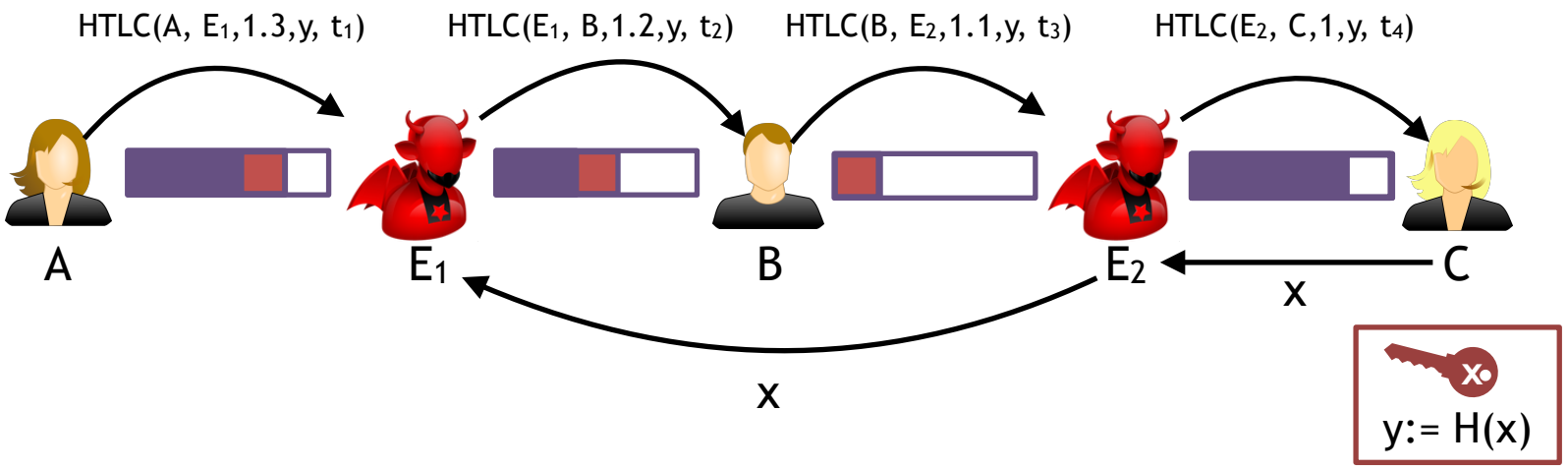
Security Issue: The Wormhole Attack



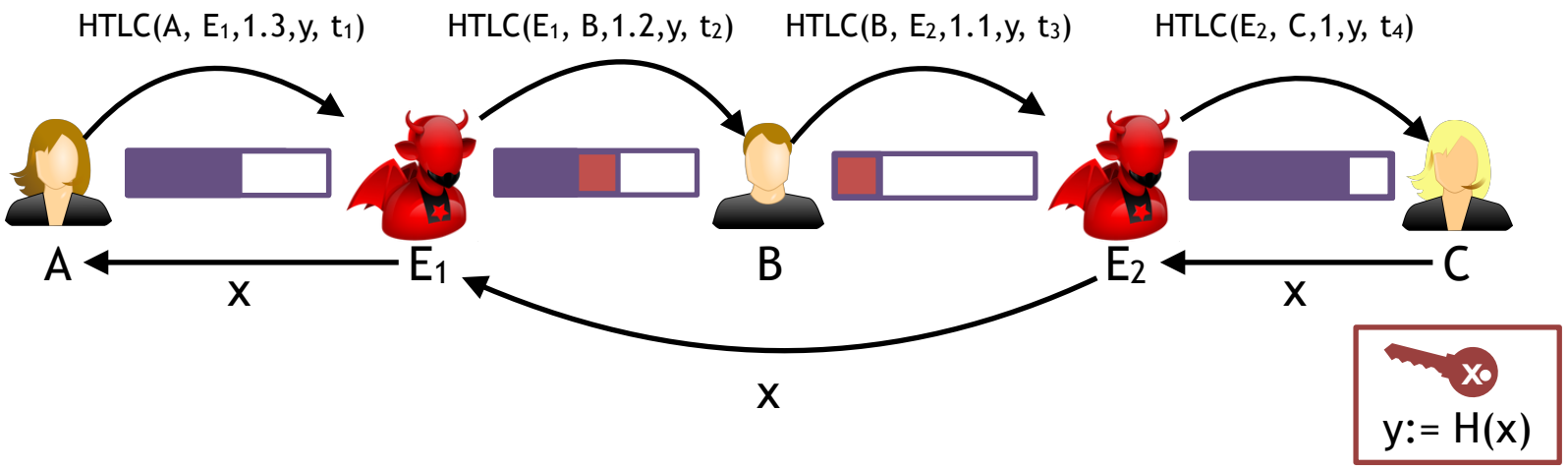
Security Issue: The Wormhole Attack



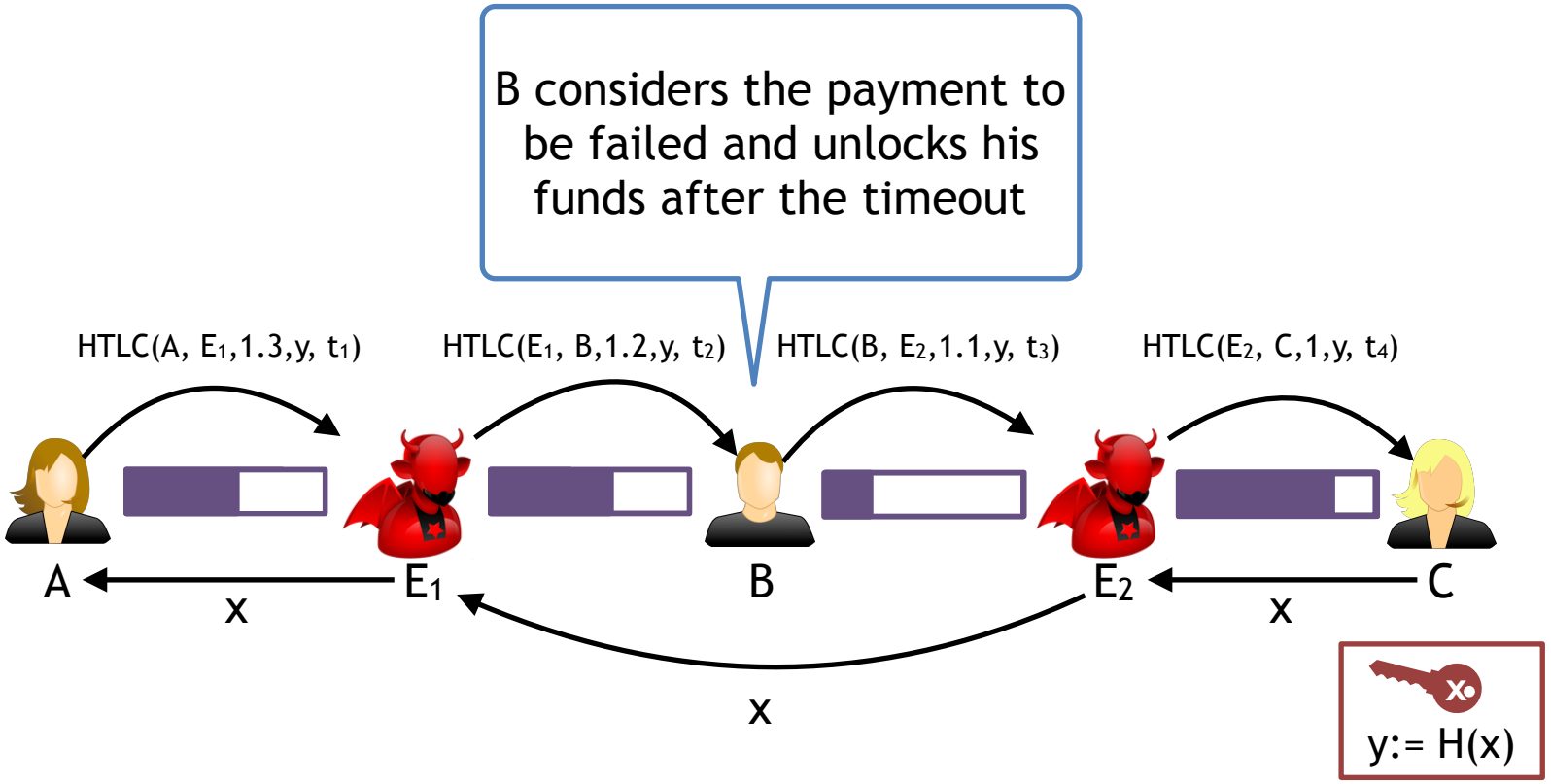
Security Issue: The Wormhole Attack



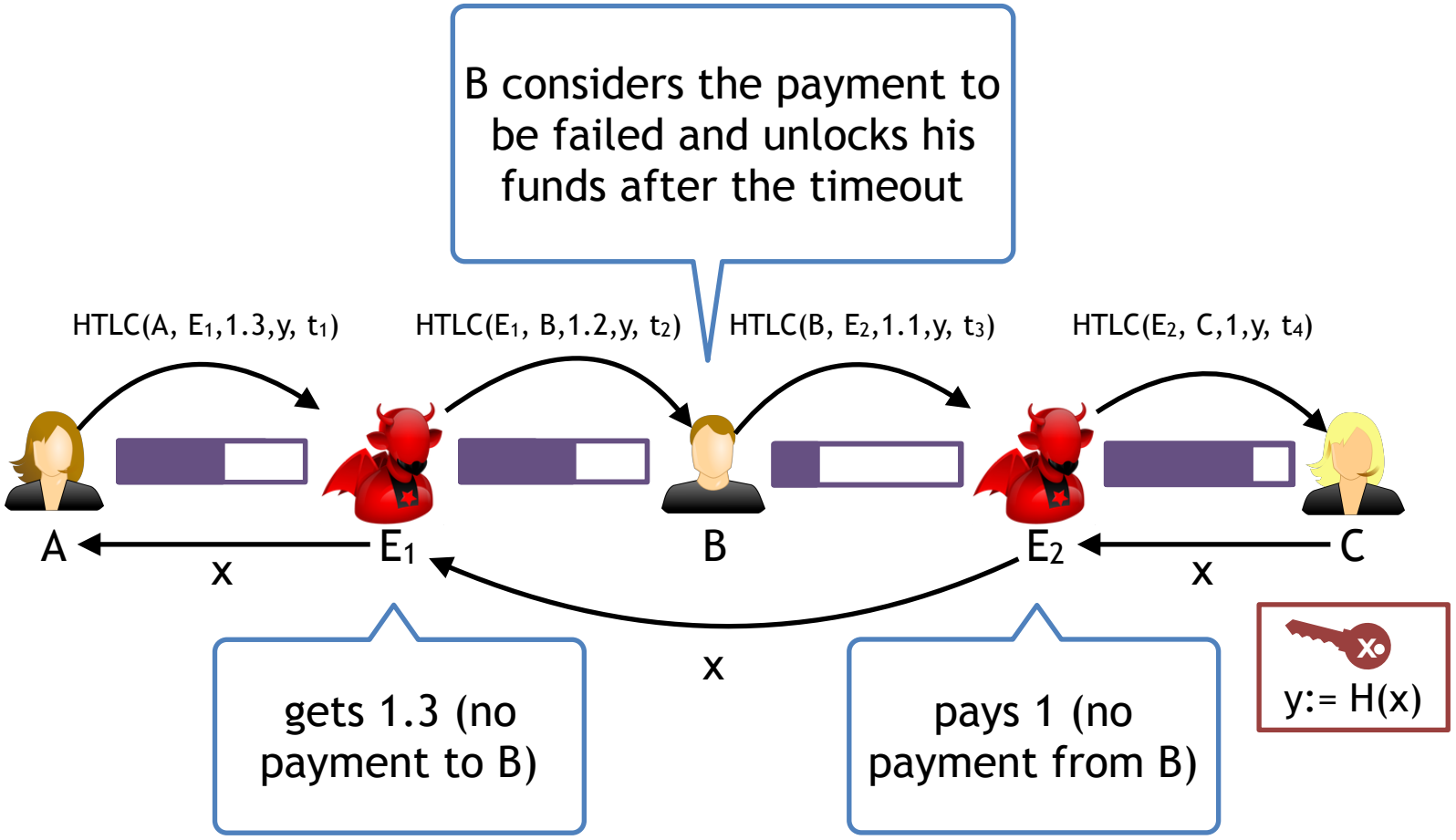
Security Issue: The Wormhole Attack



Security Issue: The Wormhole Attack

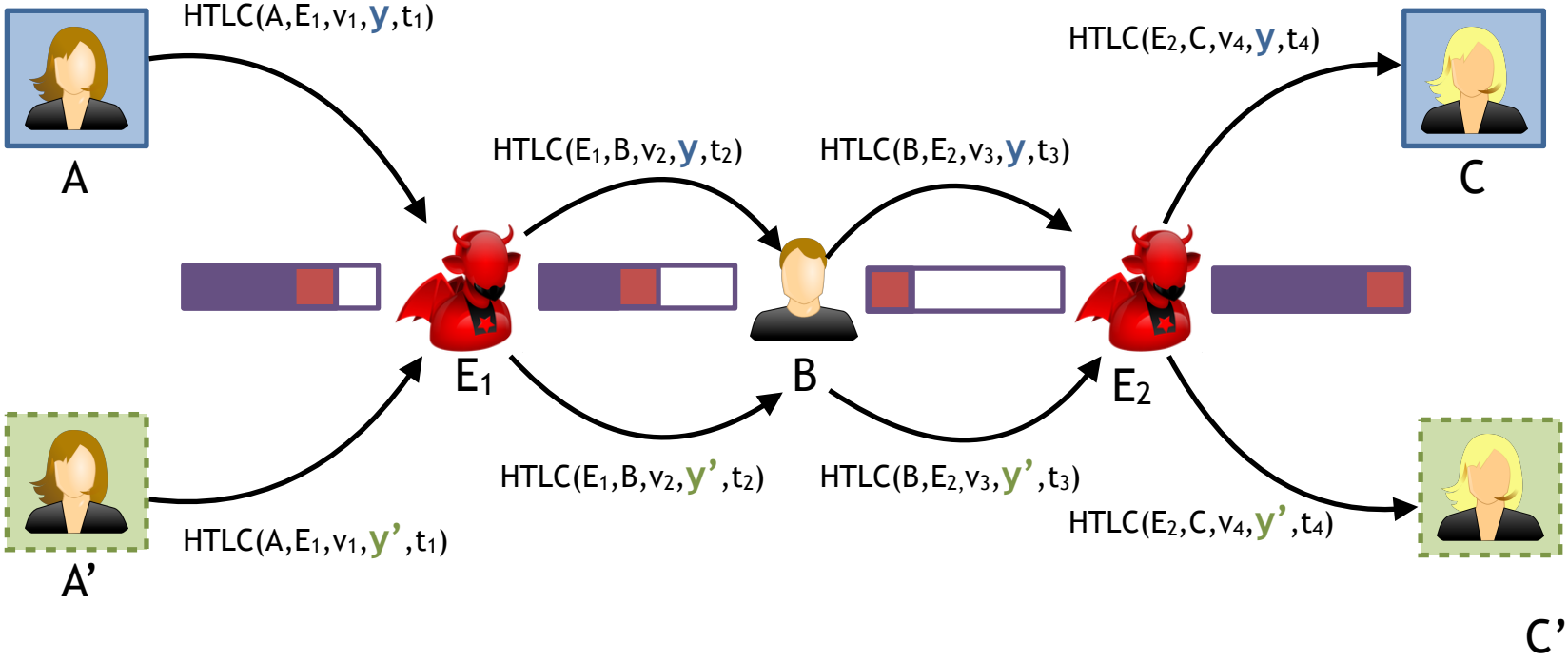


Security Issue: The Wormhole Attack



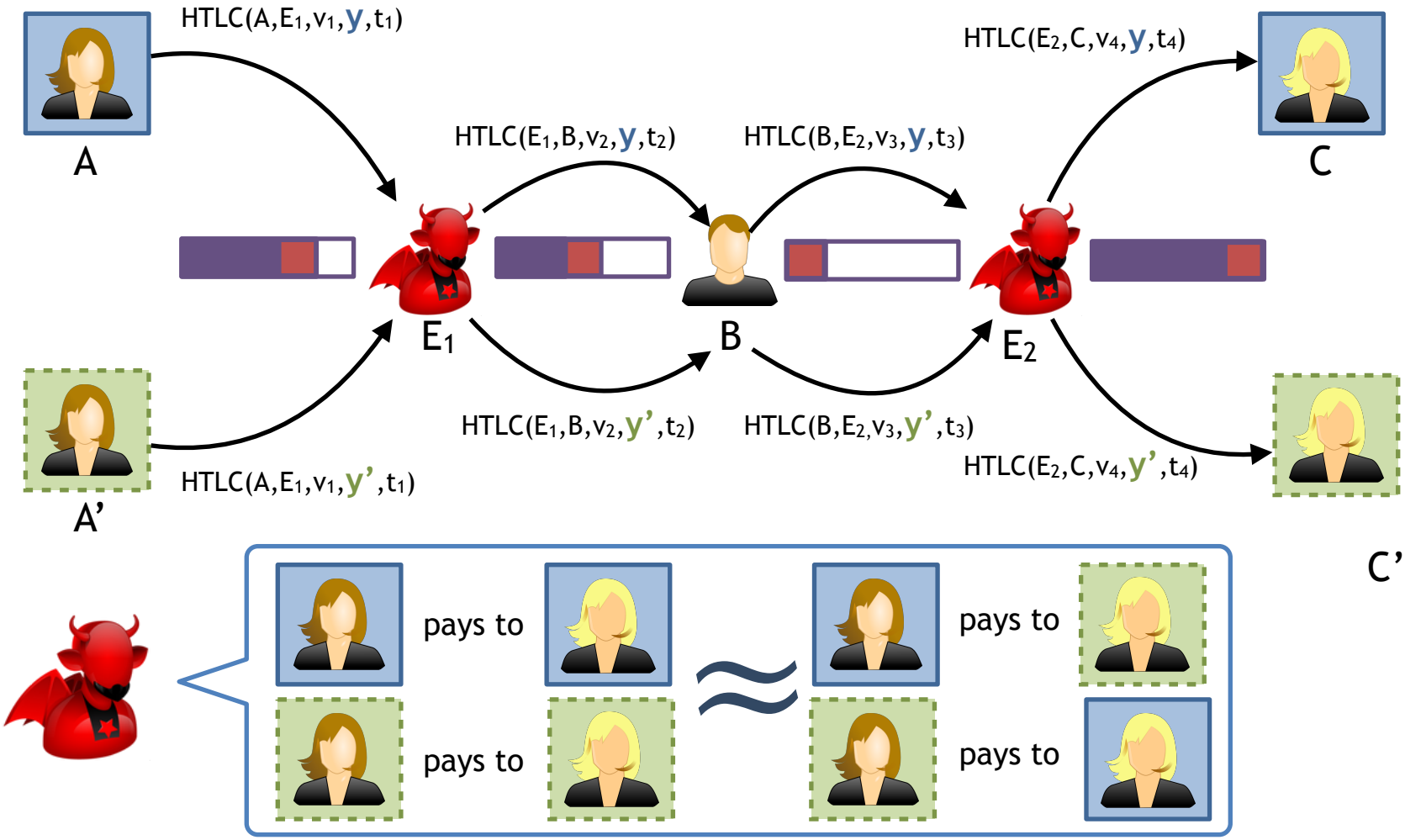
Attacker earns 0.3 BTC (own fees + B's fees)

Privacy Issues in HTLC-based Payments



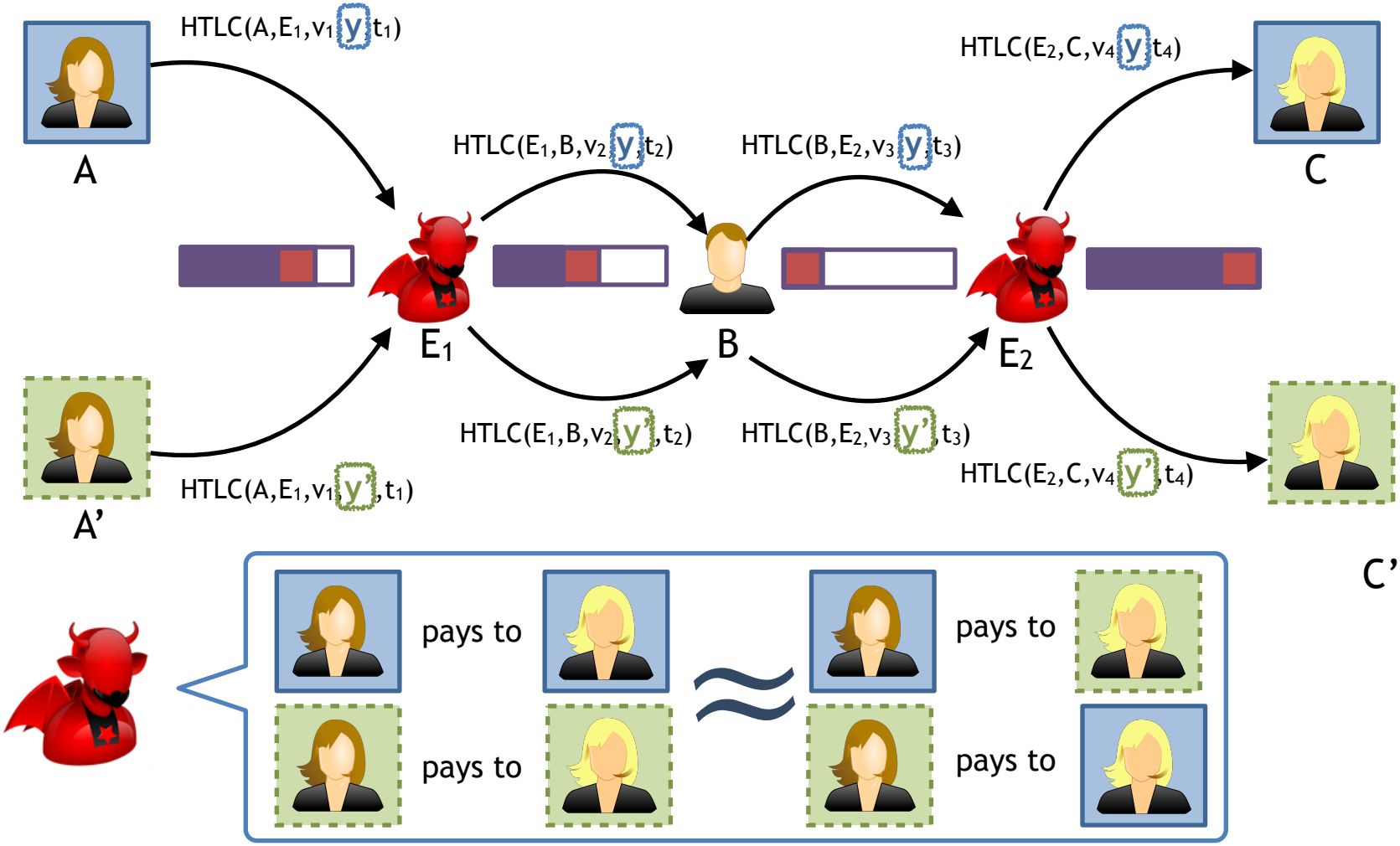
Relationship Anonymity: On-path adversaries do not learn who pays to whom

Privacy Issues in HTLC-based Payments



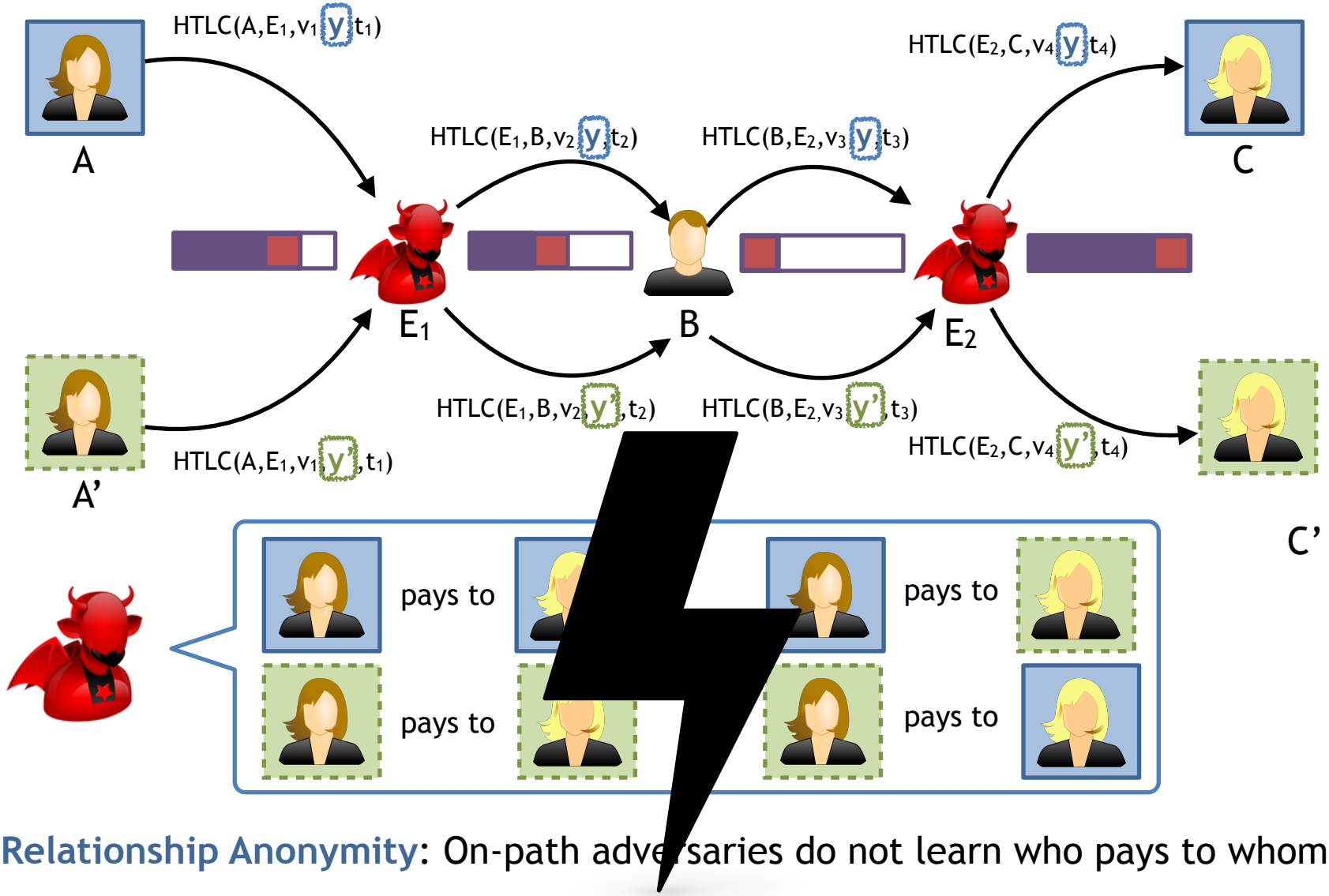
Relationship Anonymity: On-path adversaries do not learn who pays to whom

Privacy Issues in HTLC-based Payments



Relationship Anonymity: On-path adversaries do not learn who pays to whom

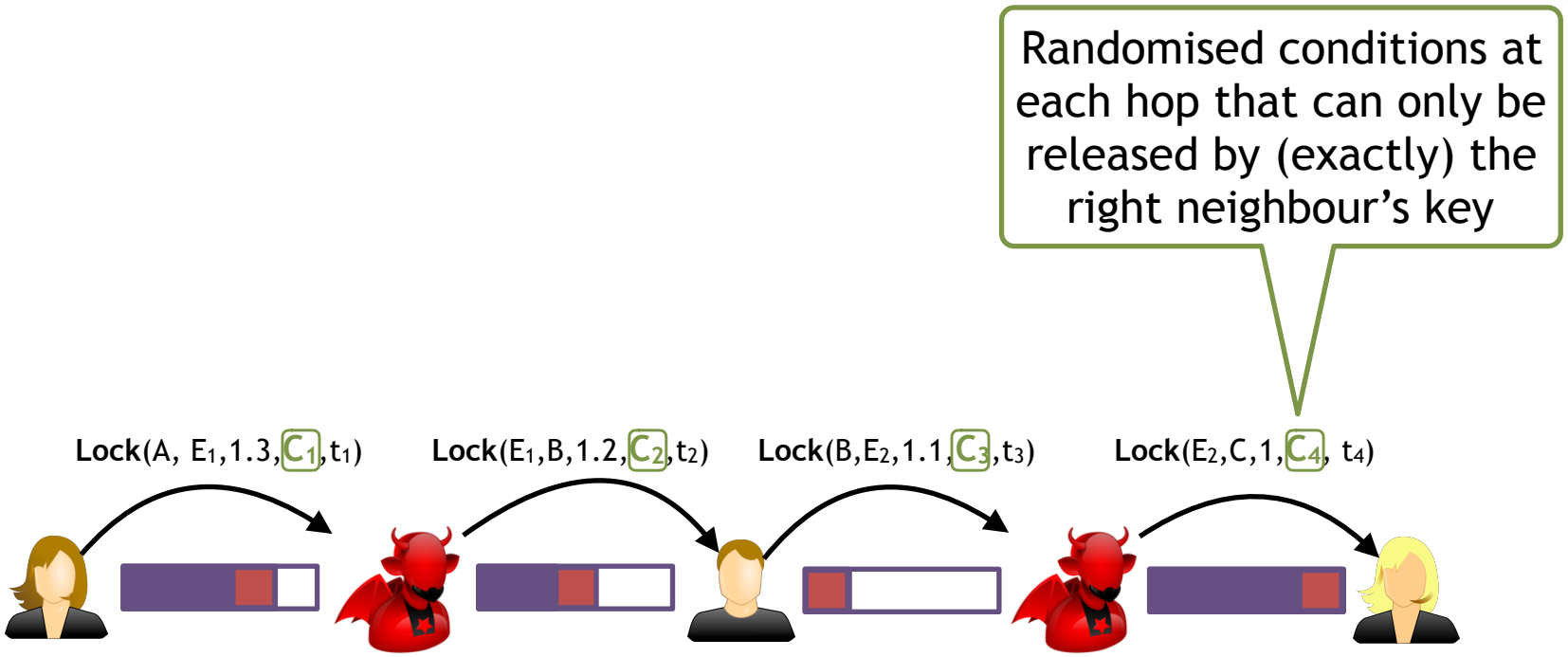
Privacy Issues in HTLC-based Payments



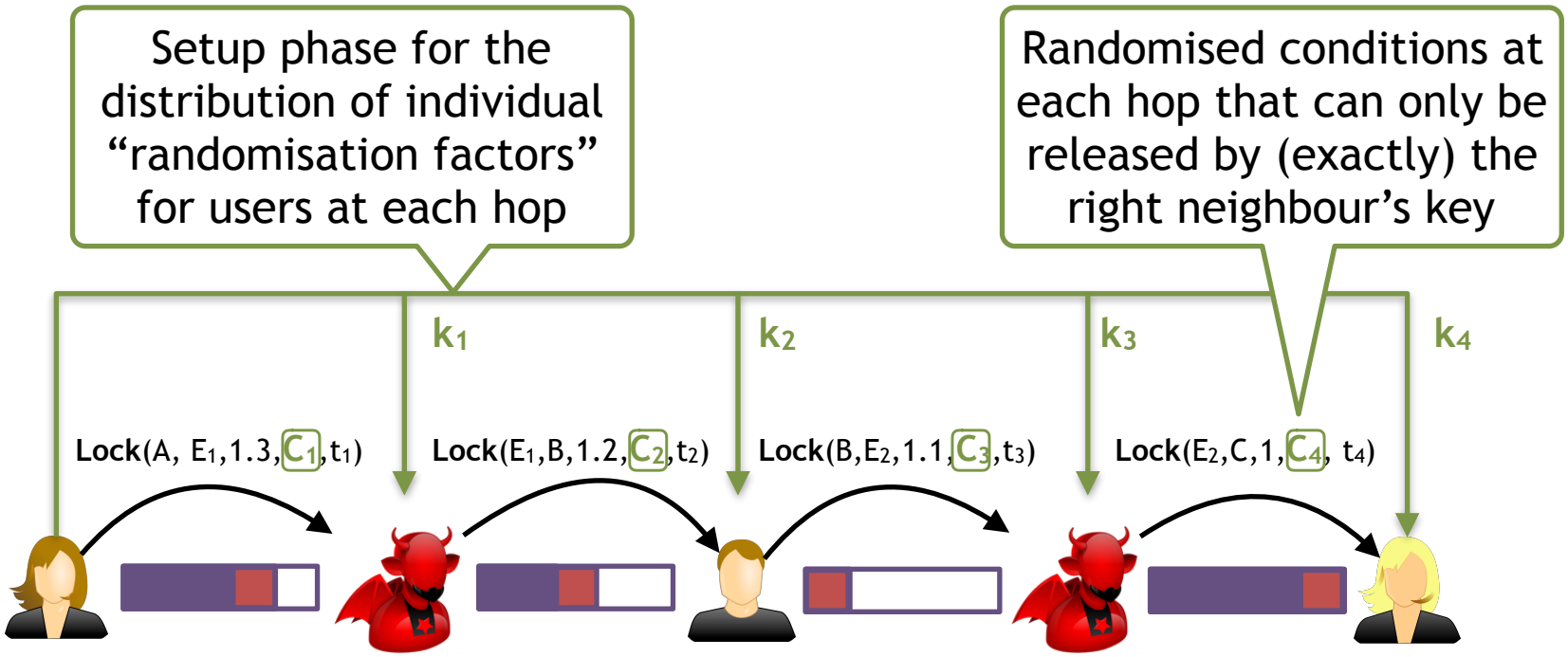
Relationship Anonymity: On-path adversaries do not learn who pays to whom

Solving Security and Privacy Issues in Payment Channel Networks

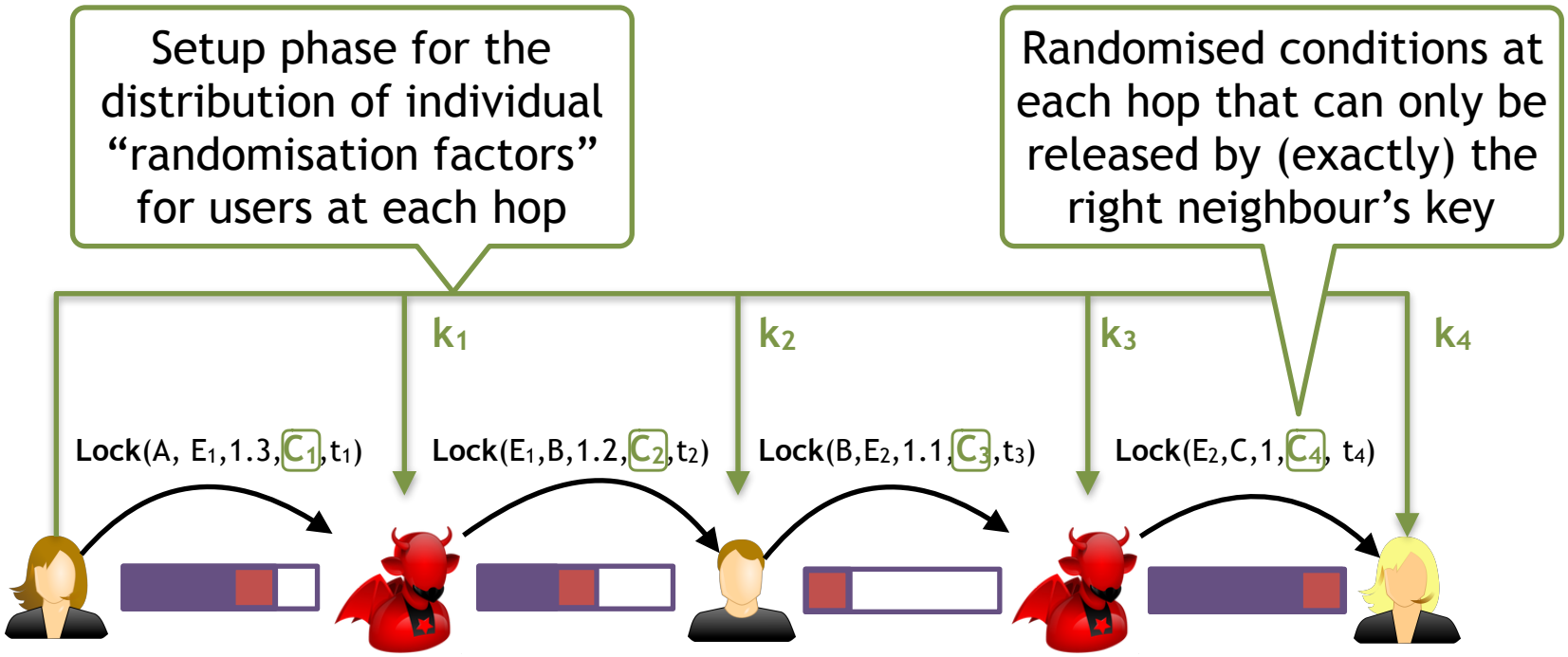
Solving Security + Privacy Issues



Solving Security + Privacy Issues



Solving Security + Privacy Issues

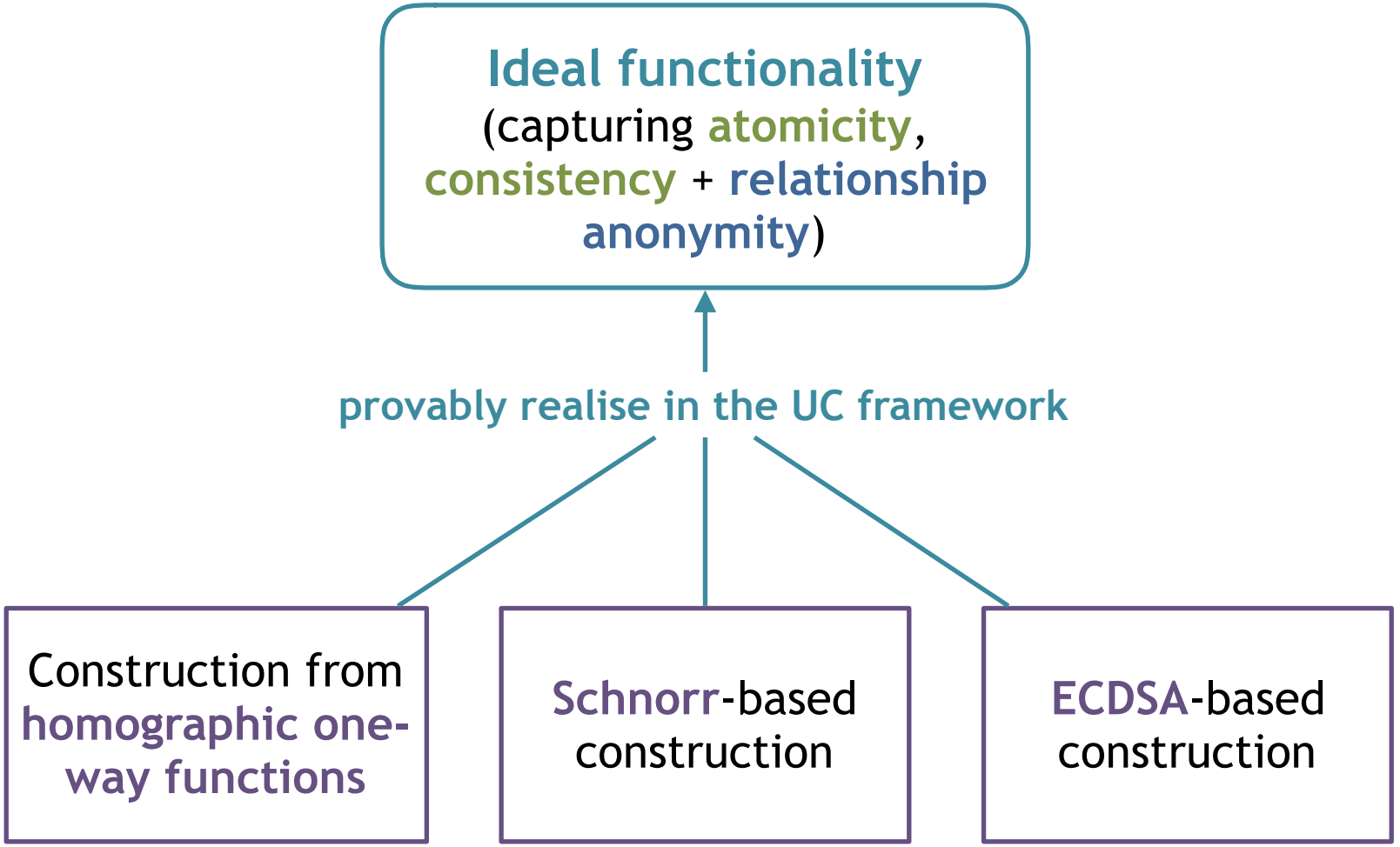


Desired Properties

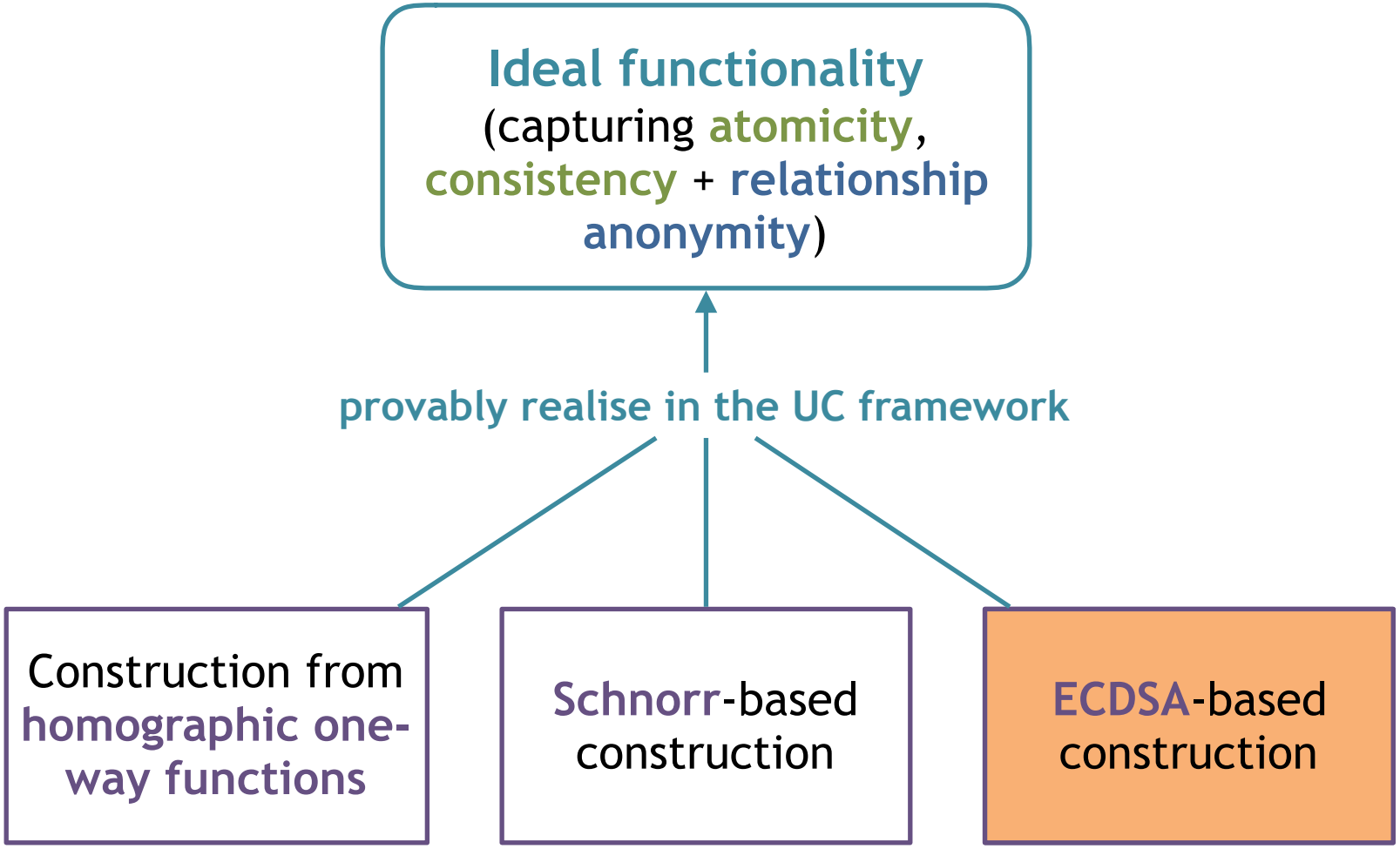
1. **Atomicity**:
   If a user's right lock gets opened, he can open his left lock
2. **Consistency**:
   A user can open his left lock only if his right lock was released
3. **Relationship Anonymity**:
   A user learns about no other participant of the payment path than his direct neighbours

No coin loss	No Wormhole Attacks	Privacy

Anonymous Multi-hop-Locks (AMHL)



Anonymous Multi-hop-Locks (AMHL)



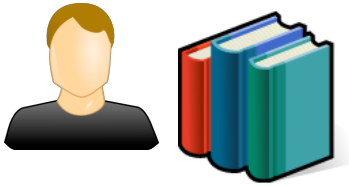
ECDSA-based Secure PCNs

Scriptless Scripts

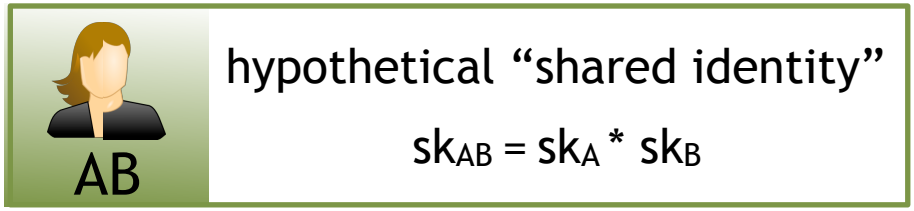
Scriptless Scripts

5

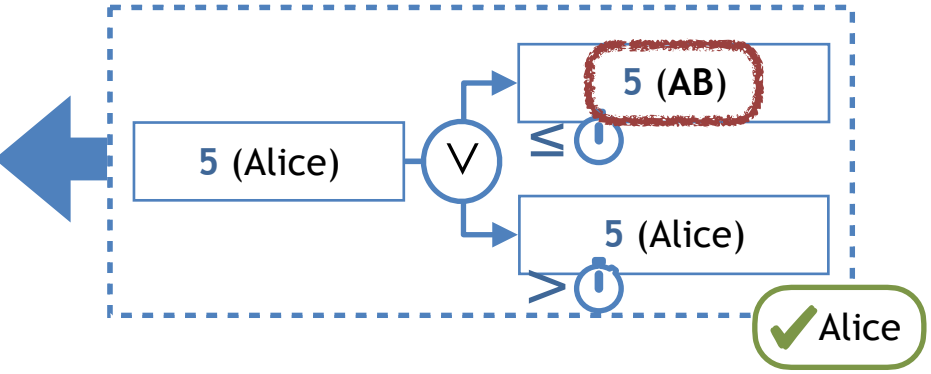
Alice
(sk_A)



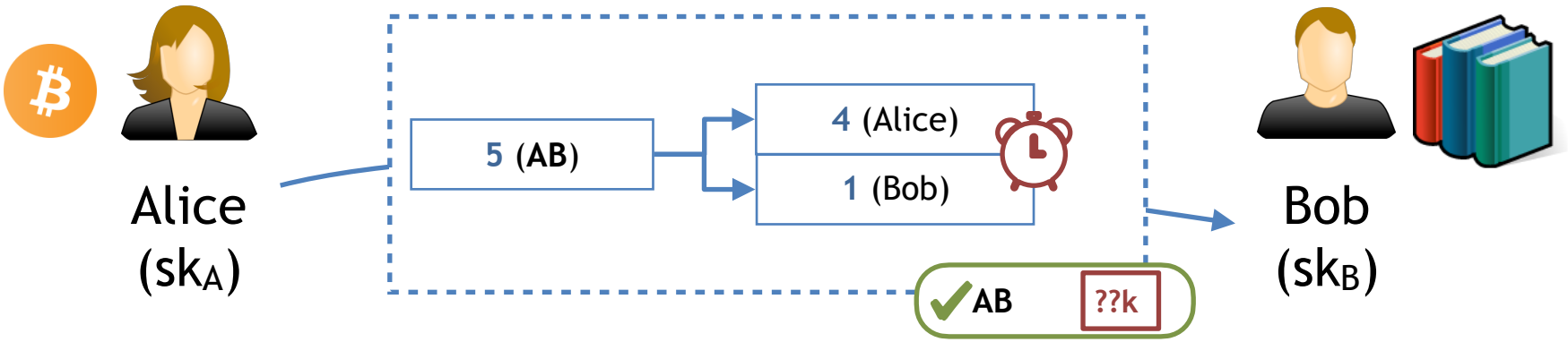
Bob
(sk_B)



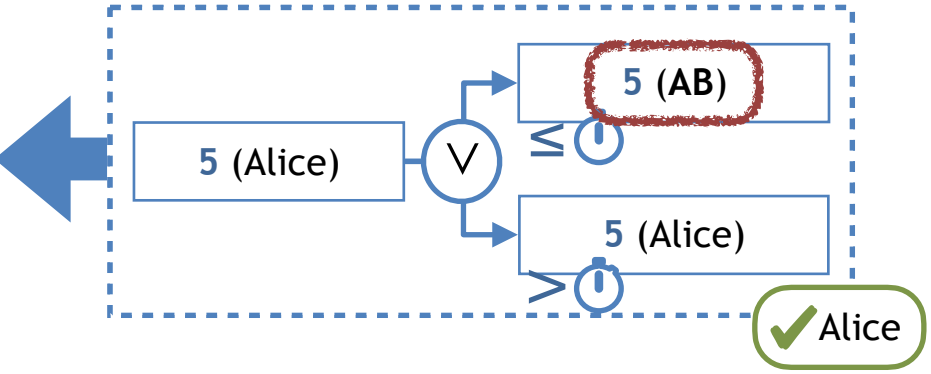
Blockchain



Scriptless Scripts



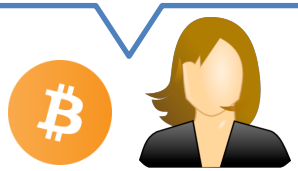
Blockchain



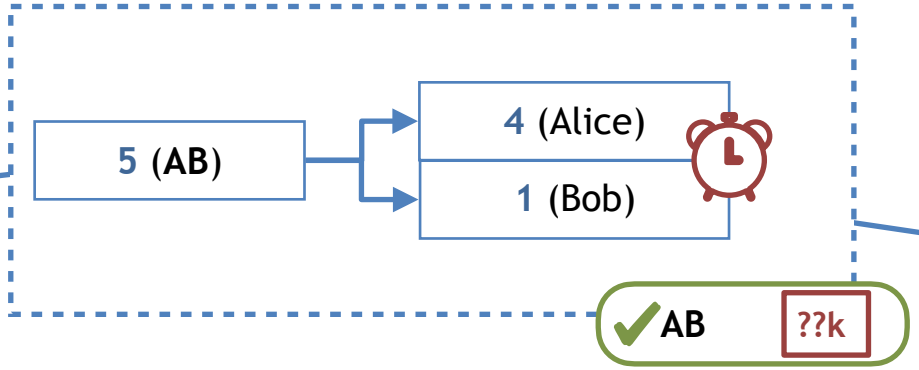
Scriptless Scripts

Alice gets sufficient information for producing a "half signature" that can be completed knowing secret **k**

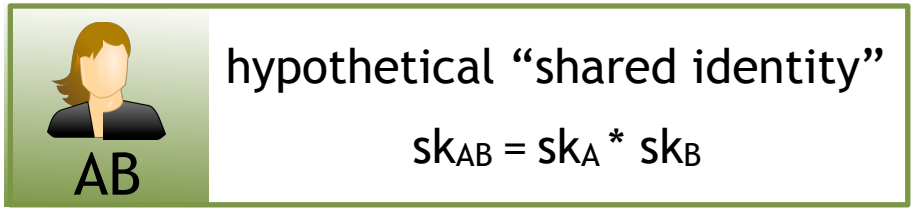
Bob gets sufficient information for checking that the "half signature" can be completed to a valid signature given **k**



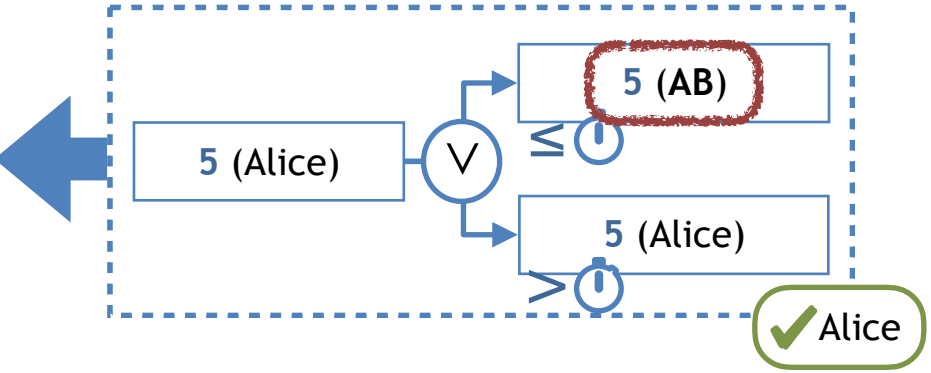
Alice
(sk_A)



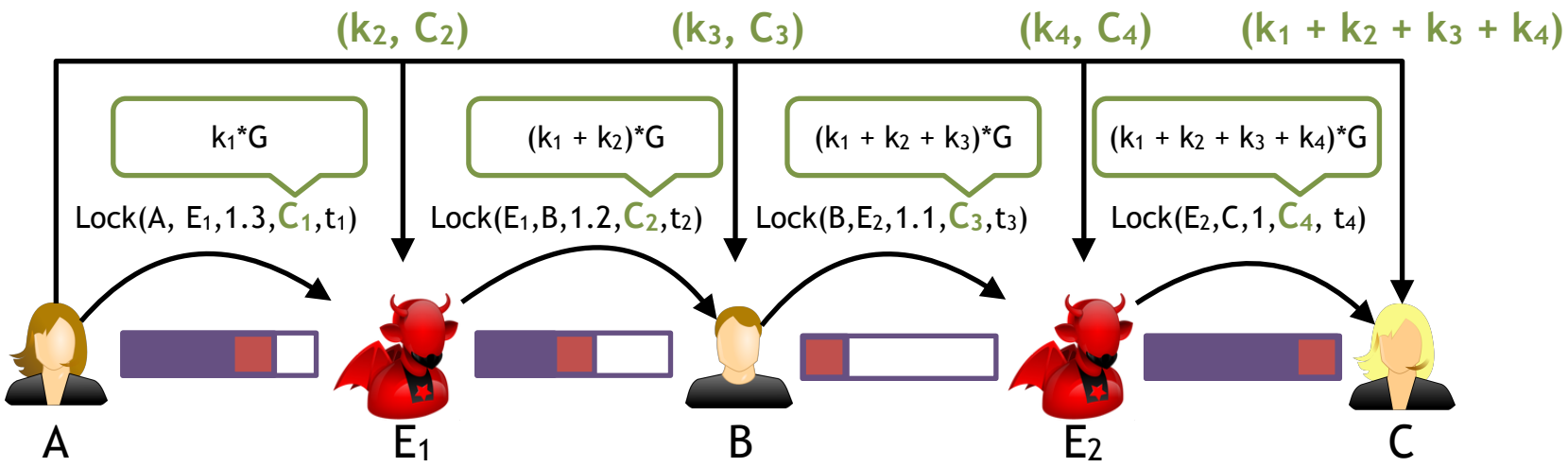
Bob
(sk_B)



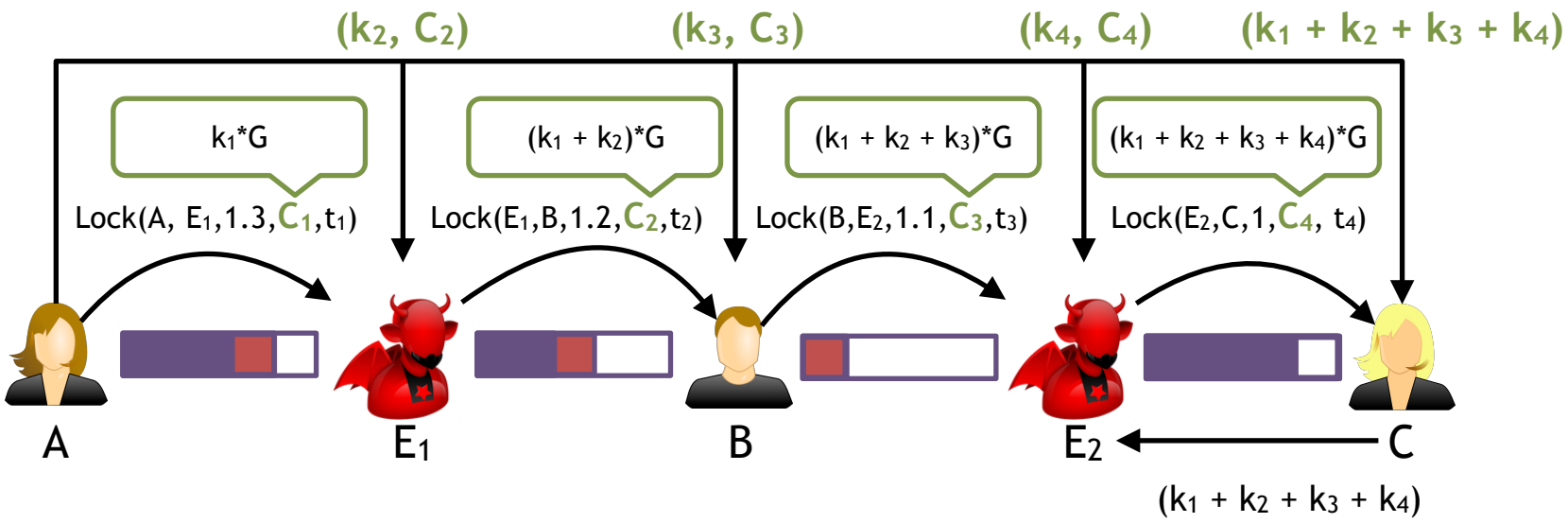
Blockchain



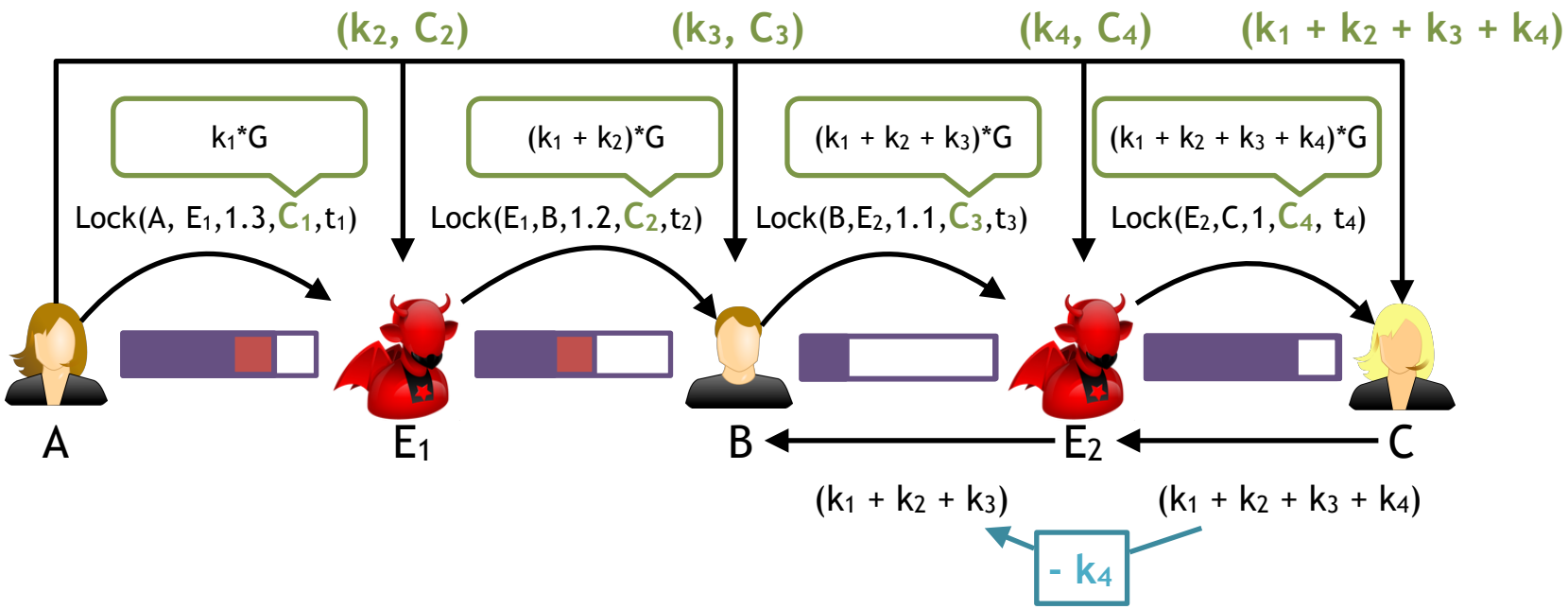
Extension to Multi-hop Locks



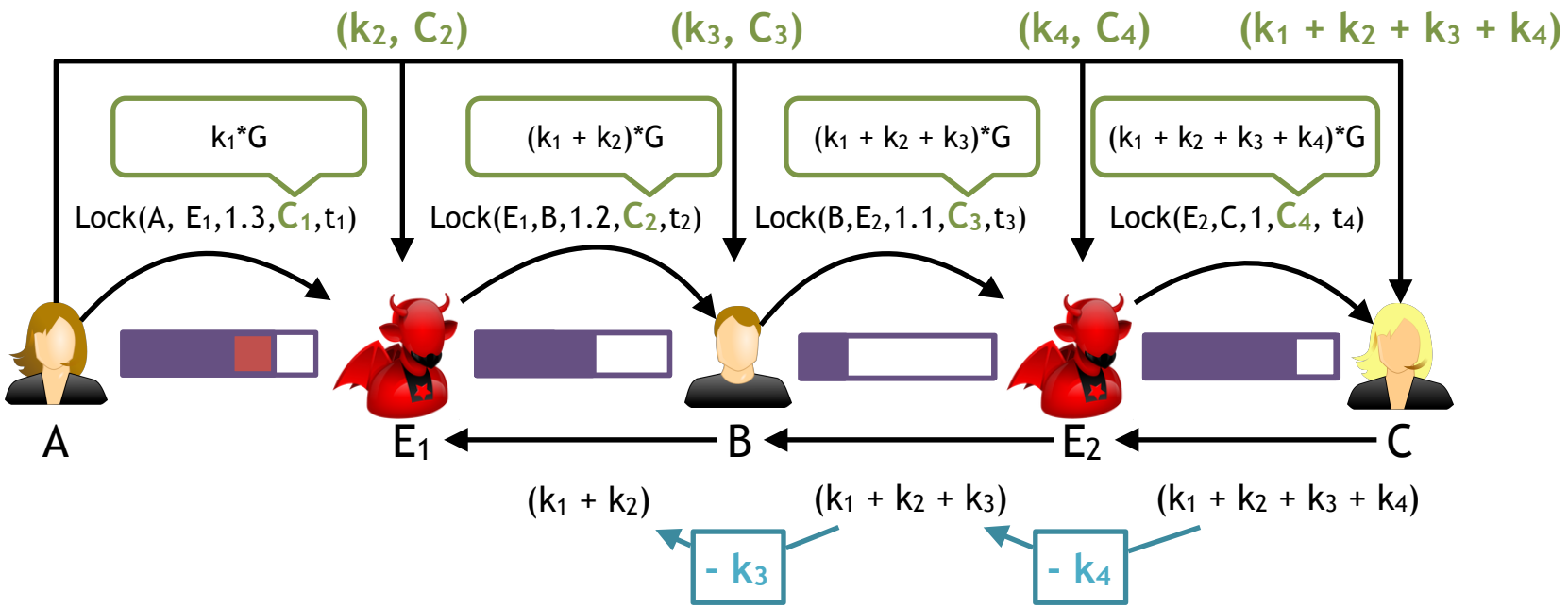
Extension to Multi-hop Locks



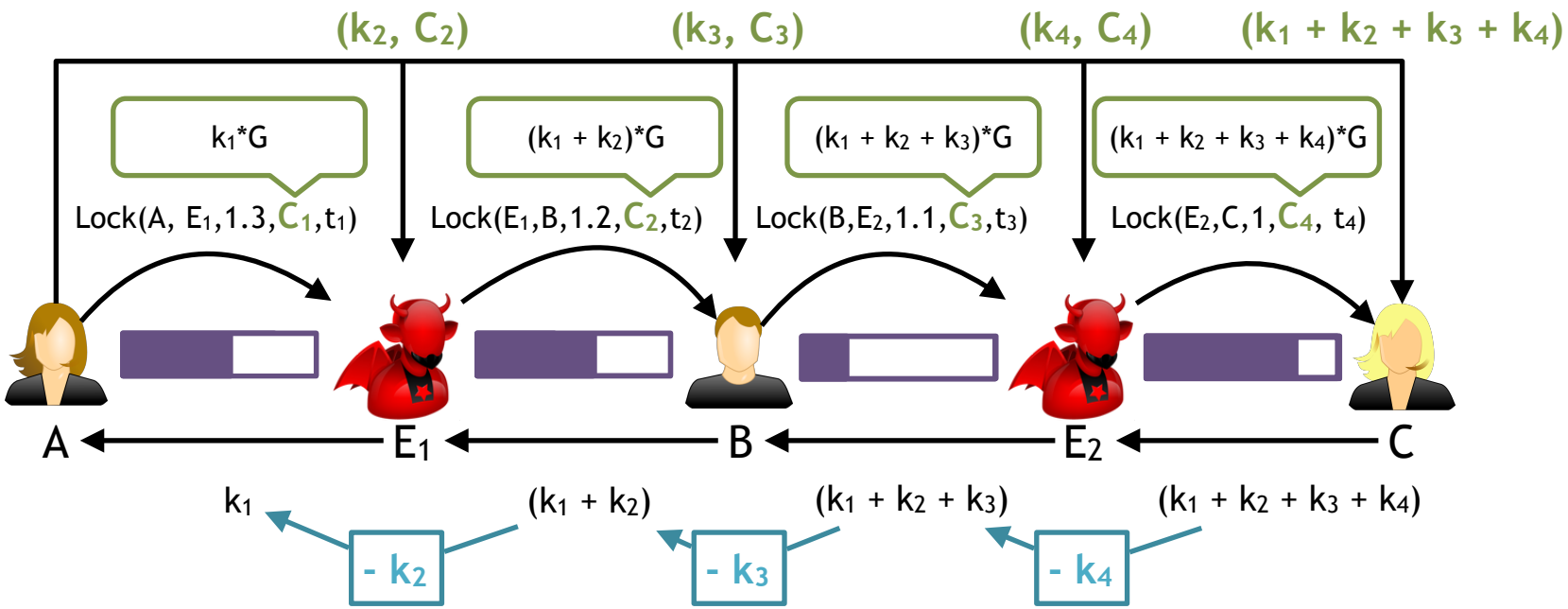
Extension to Multi-hop Locks



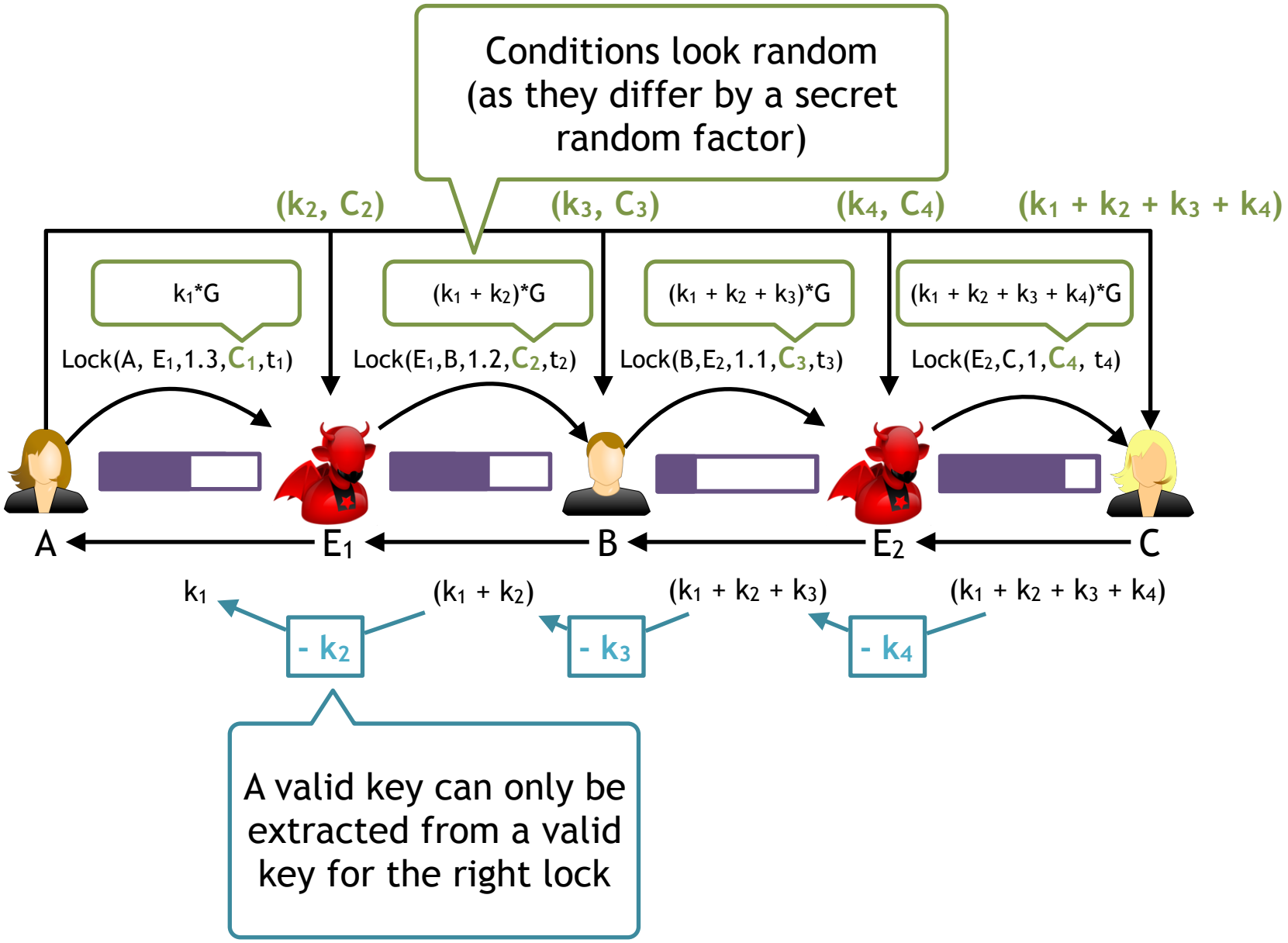
Extension to Multi-hop Locks



Extension to Multi-hop Locks



Extension to Multi-hop Locks



ECDSA-based Scriptless Lock

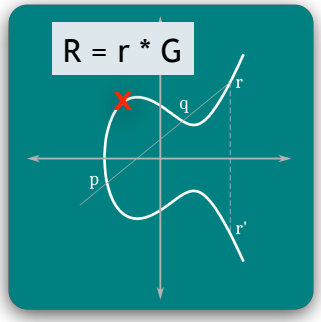
Signature w.r.t. a (public) random elliptic curve point R

secret randomness

secret key

message

σ_R = sign(r, sk, transaction)



ECDSA-based Scriptless Lock

Signature w.r.t. a (public) random elliptic curve point R

secret randomness

secret key

message

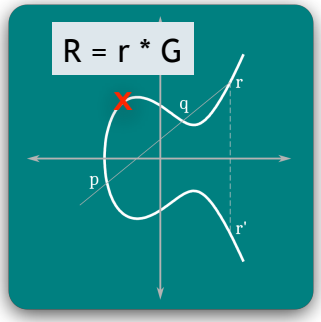
σ_R = sign(r, sk, transaction)

✔AB

$r_A * r_B * G$

$r_A * r_B$

$sk_A * sk_B$



shared signature using a shared key and randomness

ECDSA-based Scriptless Lock

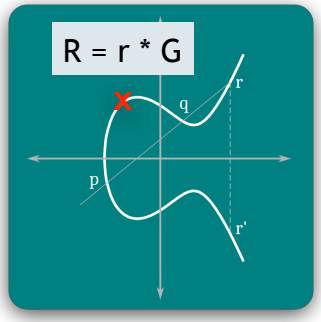
Signature w.r.t. a (public) random elliptic curve point R

secret randomness

secret key

message

σ_R = sign(r, sk, transaction)



✔AB

$r_A * r_B * G$

$r_A * r_B$

$sk_A * sk_B$

shared signature using a shared key and randomness

✔AB

$r_A * r_B * k * G$

$r_A * r_B * k$

$sk_A * sk_B$

embedding of random share (condition) **k**

ECDSA-based Scriptless Lock

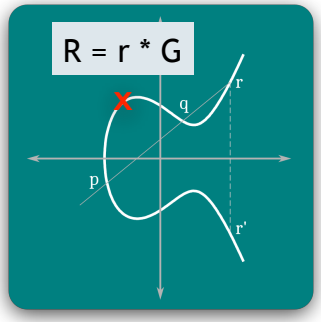
Signature w.r.t. a (public) random elliptic curve point R

secret randomness

secret key

message

$\sigma_R = \text{sign}(r, sk, \text{transaction})$



✔AB	r_A*r_B*G	r_A*r_B	sk_A*sk_B
✔AB	r_A*r_B*k*G	r_A*r_B*k	sk_A*sk_B
✔AB	r_A*r_B*k*G	r_A*r_B	sk_A*sk_B

shared signature using a shared key and randomness

embedding of random share (condition) k

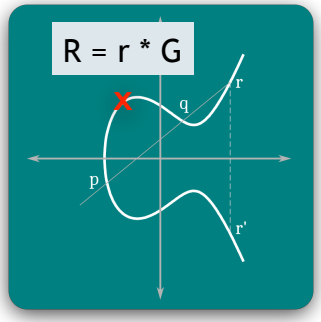
“half signature” without k but still with respect to r_A*r_B*k*G

ECDSA-based Scriptless Lock

Signature w.r.t. a (public) random elliptic curve point R

secret randomness | secret key | message

$\sigma_R = sign(r, sk, transaction)$



✓AB	r_A*r_B*G	r_A*r_B	sk_A*sk_B
✓AB	r_A*r_B*k*G	r_A*r_B*k	sk_A*sk_B
✓AB	r_A*r_B*k*G	r_A*r_B	sk_A*sk_B

shared signature using a shared key and randomness

embedding of random share (condition) k

“half signature” without k but still with respect to r_A*r_B*k*G

Lock Protocol

(sk_A, r_A)

$C=k*G$, transaction

…

“1/3” signature $\sigma_{R,B}$

“1/3” signature $\sigma_{R,A}$

(sk_B, r_B)

✓AB

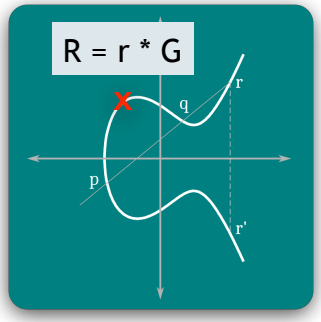
✓AB

ECDSA-based Scriptless Lock

Signature w.r.t. a (public) random elliptic curve point R

secret randomness | secret key | message

σ_R = sign(r, sk, transaction)



✓AB	r_A*r_B*G	r_A*r_B	sk_A*sk_B
✓AB	r_A*r_B*k*G	r_A*r_B*k	sk_A*sk_B
✓AB	r_A*r_B*k*G	r_A*r_B	sk_A*sk_B

shared signature using a shared key and randomness

embedding of random share (condition) k

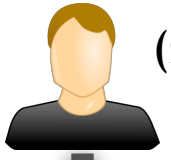
“half signature” without k but still with respect to r_A*r_B*k*G

Lock Protocol

(sk_A, r_A)

C=k*G, transaction

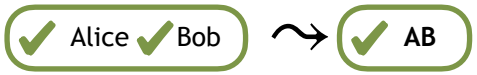
Hard for ECDSA as σ_R has a non-linear structure



(sk_B, r_B)

Properties/Evaluation

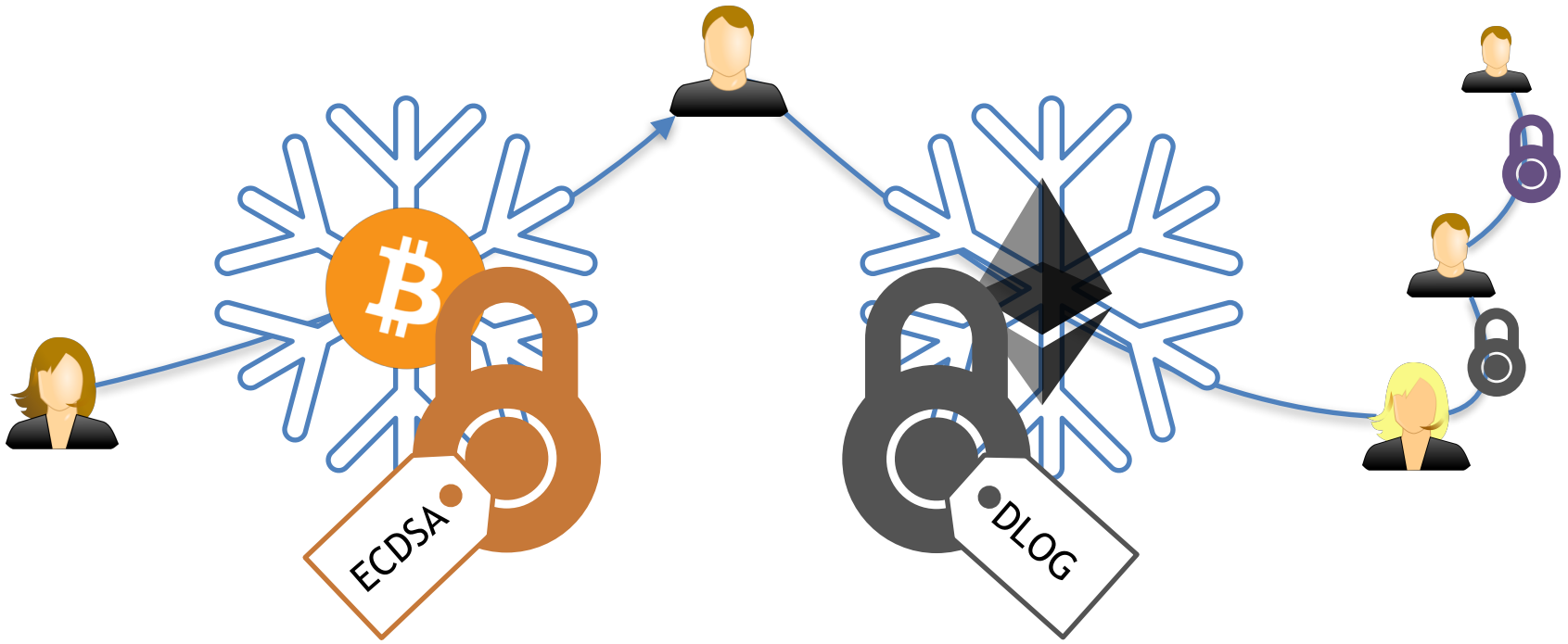
- Security and Privacy proven in the UC Framework
- Compatible with Bitcoin and current PCNs
  - ✓ Implemented in the Lightning Network (https://github.com/cfromknecht/tpec)
- Reduces transaction size for conditional payments
  - ✓ Encoding of condition within signature 

- Makes settlement transactions indistinguishable from regular ones (Fungibility) 

- Little overhead:
  - ✓ < 500 bytes communication
  - ✓ few ms computation

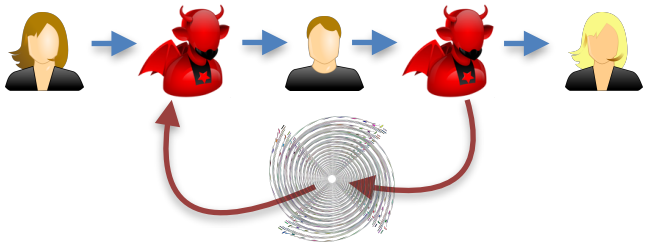
Interoperability

- AMHLs are suitable for cross-currency usage
  - even with different primitive instantiations
  - ✓ Inter-currency payment channels
  - ✓ Atomic swaps

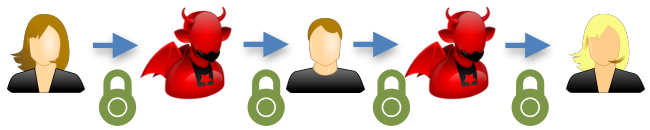


Summary

The **Wormhole Attack**:
A novel attack on Payment Channel Network Security



AMHLs: A **new primitive** for secure + anonymous Payment Channel Networks

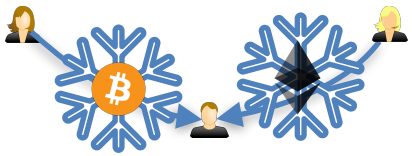


Concrete **constructions** of AMHLs that

… are efficient

… got implemented in Bitcoin's Lightning Network

… enable inter-blockchain Payment Channels

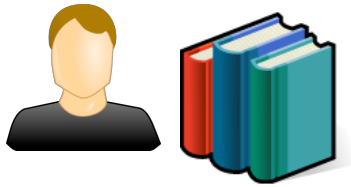


Additional Material

HTLC in practice

Alice

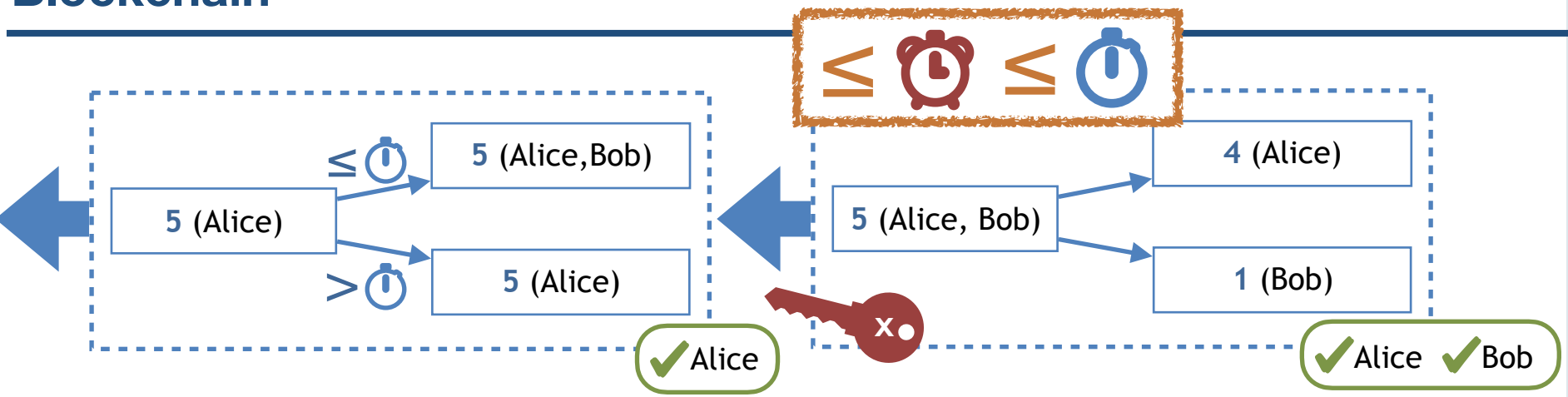
HTLC (Alice, Bob, 1, y, ⏰):
Alice pays Bob 1 BTC iff Bob shows some x such that H(x) = y before ⏰



Bob

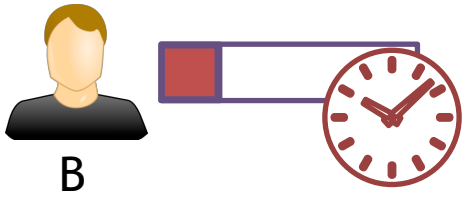
Transaction can only be added
1) Before the expiration of the payment channel
2) Before the expiration of the HTLC
3) when providing the pre-image x

Blockchain



Implications of the Wormhole Attack

- Collateral cost: Honest intermediaries' coins are locked (cannot be used in a successful payment)



- Attacked intermediaries cannot distinguish between an attack and a failed payment

➡ Destroys the incentive for intermediaries to participate in multi-hop payments at all

Properties of Multi-hop-lock-based PCN

Atomicity

Balance Security

Consistency

Prevents Wormhole Attacks

Relationship Anonymity

(limited) Privacy

Minimal requirements

which (cryptographic) constructs are needed for implementing the Locks

Compatibility + Interoperability

Communication

Which amount of communication is required for building payment paths from locks

Performance

Blockchain Effects

How do the locks influence the transactions visible on the Blockchain

Fungibility + Blockchain growth

Properties of the Different Constructions

	Current PCN	OWH-based PCN	Schnorr-based PCN	ECDSA-based PCN
Atomicity	✔	✔	✔	✔
Consistency	✘	✔	✔	✔
Privacy	✘	✔	✔	✔
Compatibility/ Interoperability	✘	partly	partly	✔
Fungibility + reduced transaction size	✘	✘	✔	✔

Scriptless Locks

signature with respect to message m and (random) point (R_x,R_y) on a elliptic curve

Schnorr Signature

m

$$(R_x, R_y) = r * G$$

$$r + sk * m$$

ECDSA Signature

$$\frac{R_x * sk + m}{r}$$

simple linear combination of additive key and random shares

Signature of shared key and randomness

complex combination (x-coordinate, multiplicative shares, inverse)

$$= (r_A + r_B) * G$$

$$= (r_A * r_B) * G$$

$$(r_A + r_B) + (sk_A + sk_B) * m$$

$$\frac{R_x * sk_A * sk_B + m}{r_A * r_B}$$

Embedding of arbitrary random shares (conditions)

$$= (r_A + r_B + k) * G$$

$$= (r_A * r_B * k) * G$$

$$(r_A + r_B + k) + (sk_A + sk_B) * m$$

$$\frac{R_x * sk_A * sk_B + m}{r_A * r_B * k}$$

$$\sigma_{sk_A * sk_B}^{(r_A * r_B * k) * G}(m, r_A * r_B * k)$$

ECDSA-based Lock

For condition $C = k*G$ compute signature $\sigma^{/\{k\}}$ s. t. $\frac{\sigma^{/\{k\}}}{k} = \sigma_{sk_{AB}}^{r_A * r_B * C}(m, r_A * r_B * k)$

pk_{AB} m

sk_A

Alice

AB

$(sk_{AB}, pk_{AB}) = (sk_A * sk_B, sk_A * sk_B * G)$

pk_{AB} m

Alice

sk_A

sk_B

Bob

$r_A, R_B = r_B*G$

$R = (R_x, R_y) = r_A*r_B*C$

joint randomness generation

$r_B, R_A = r_A*G$

$R = (R_x, R_y) = r_A*r_B*C$

$= r_A*r_B*k*G$

Alice $\sigma^{/\{k, r_A\}}$

Alice $\sigma^{/\{k, r_A\}}$:= Alice $\sigma_{sk_{AB}}^{R}(m, r_B)$

$\sigma^{/\{k\}} := \frac{\sigma^{/\{k, r_A\}}}{r_A}$

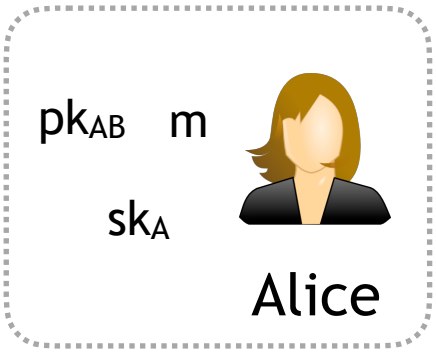
$= \sigma_{sk_{AB}}^{R}(m, r_A * r_B)$

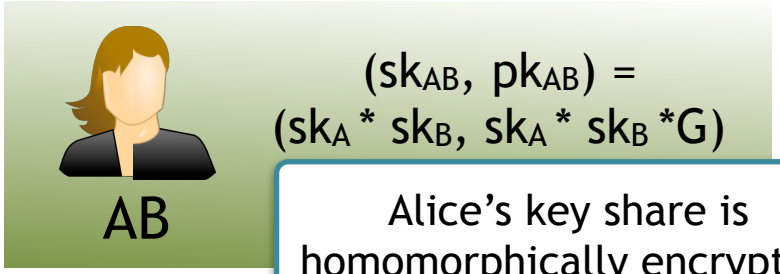
$\sigma^{/\{k\}}$

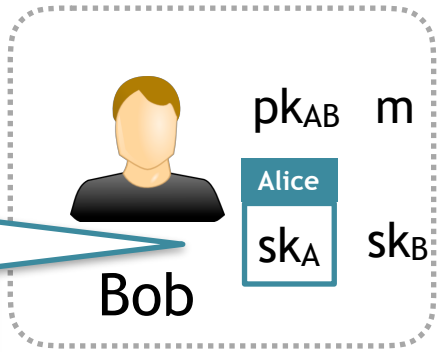
requires sk_A and hence can only be performed under homomorphic encryption for Alice

ECDSA-based Lock

For condition $C = k*G$ compute signature $\sigma^{/\{k\}}$ s. t. $\frac{\sigma^{/\{k\}}}{k} = \sigma_{sk_{AB}}^{r_A * r_B * C}(m, r_A * r_B * k)$







$r_A, R_B = r_B*G$

$R = (R_x, R_y) = r_A*r_B*C$

$= r_A*r_B*k*G$

$r_B, R_A = r_A*G$

$R = (R_x, R_y) = r_A*r_B*C$

Alice $\sigma^{/\{k, r_A\}}$

Alice $\sigma^{/\{k, r_A\}}$:= Alice $\sigma_{sk_{AB}}^{R}(m, r_B)$

$\sigma^{/\{k\}} := \frac{\sigma^{/\{k, r_A\}}}{r_A}$

$= \sigma_{sk_{AB}}^{R}(m, r_A * r_B)$

$\sigma^{/\{k\}}$

requires sk_A and hence can only be performed under homomorphic encryption for Alice