Schnorr,
Adaptor Sigs
and Statechains

Ruben Somsen

What will be covered?

- 5 min recap of Statechains
- A crash course on Schnorr
- Adaptor Signatures
- Atomic transfers in Statechains

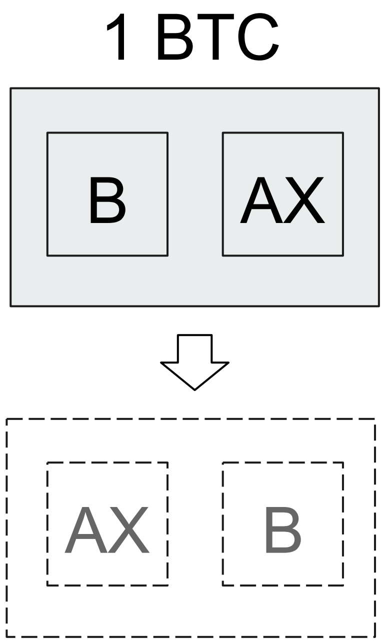
Statechains
5 min recap

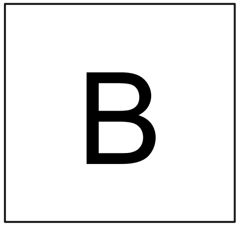
Statechains

- 2-of-2 channel between “Statechain entity” and users
- Transfer entire UTXOs (one chain each)
- More secure thanks to on-chain redemption
- Minimum complexity, contracts enforced on-chain

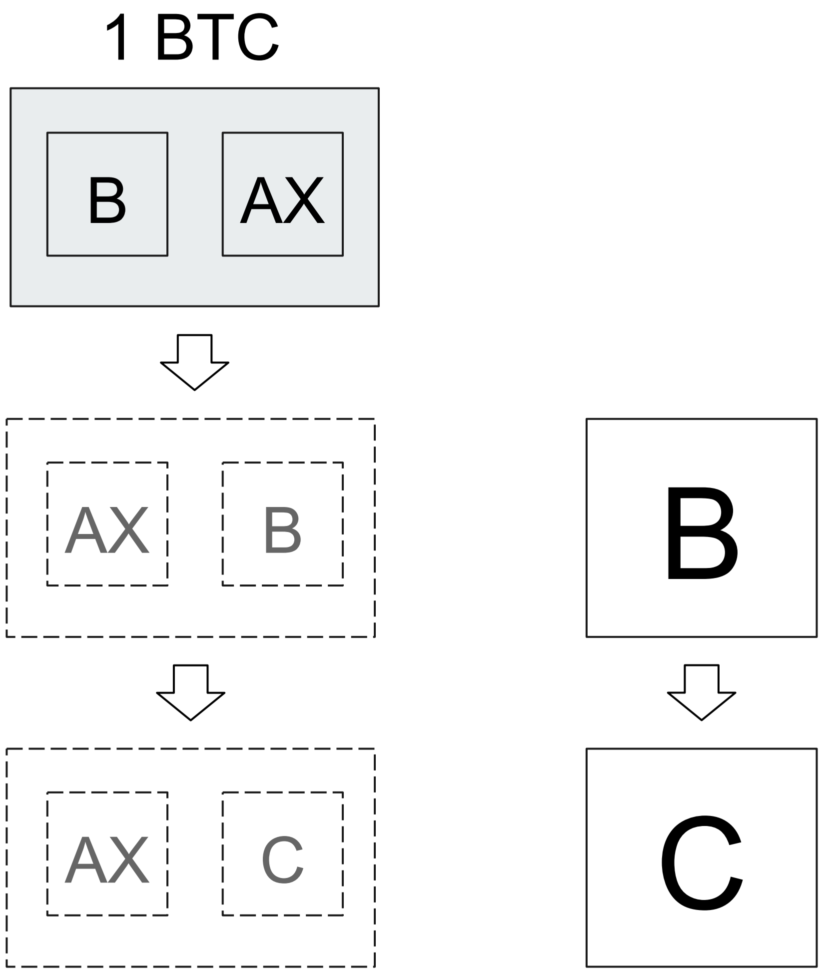
Bitcoin

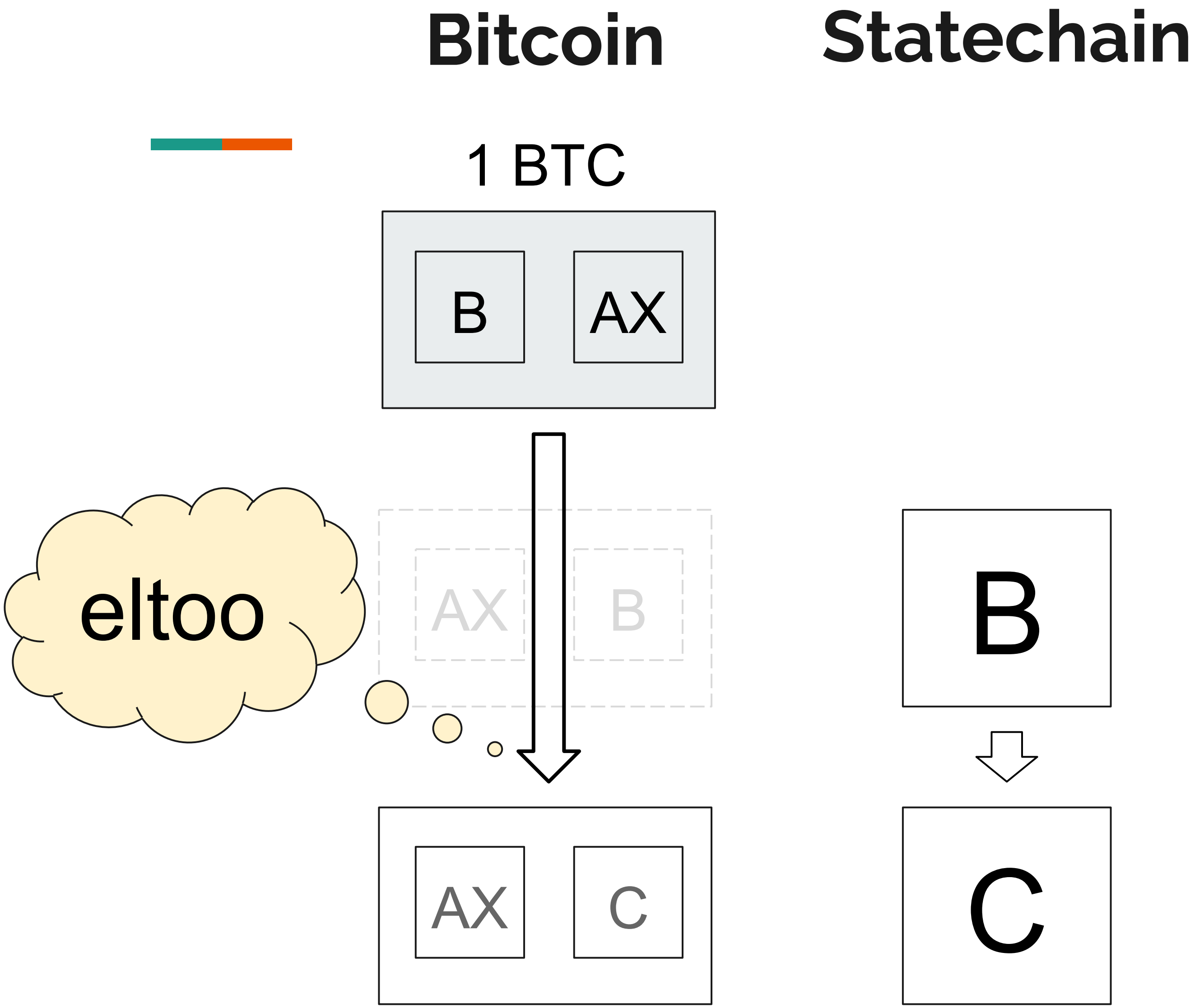
Statechain





Bitcoin Statechain

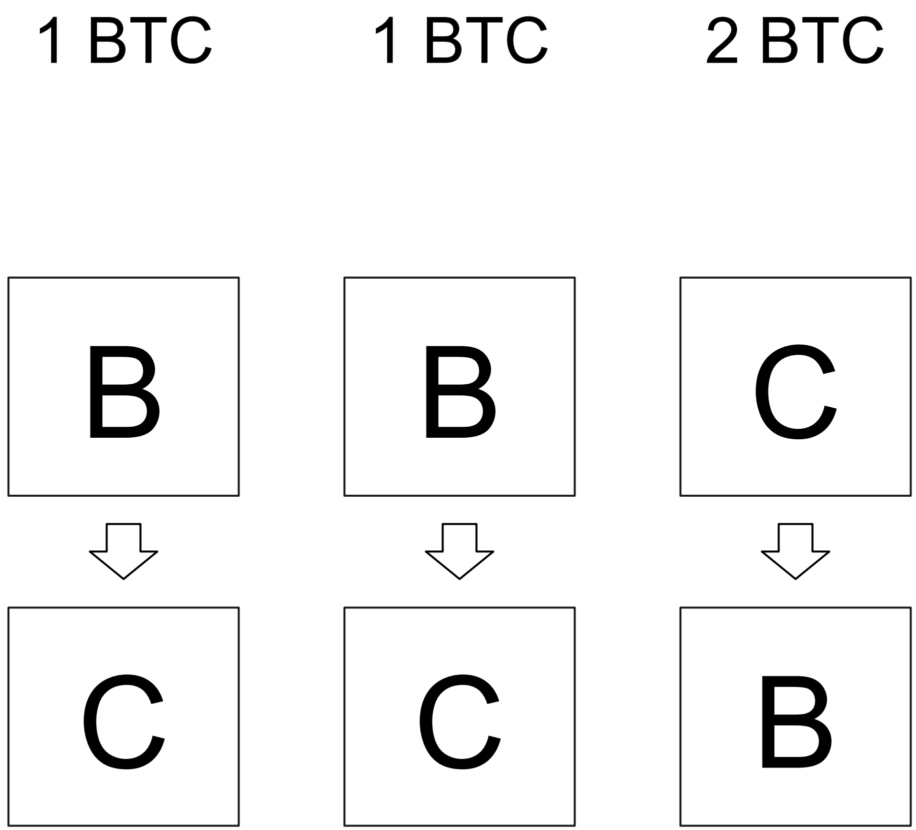



Bitcoin
Statechain
1 BTC
B
AX
eltoo
AX
B
B
AX
C
C

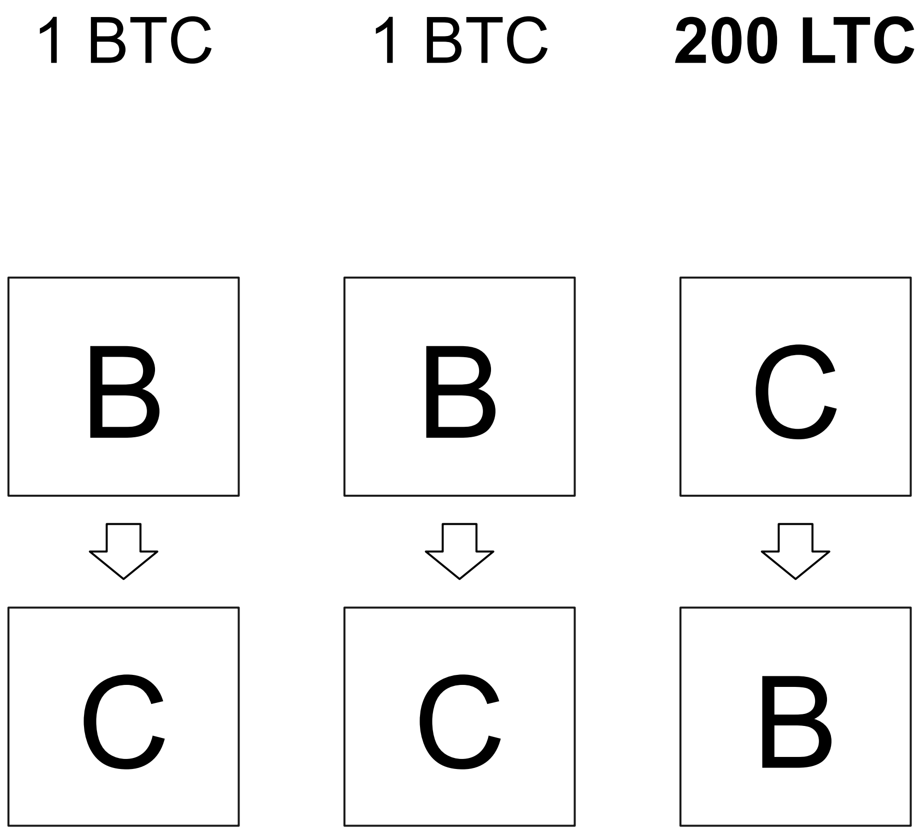
Swapping to smaller amounts

1 BTC	1 BTC	2 BTC
B	B	C

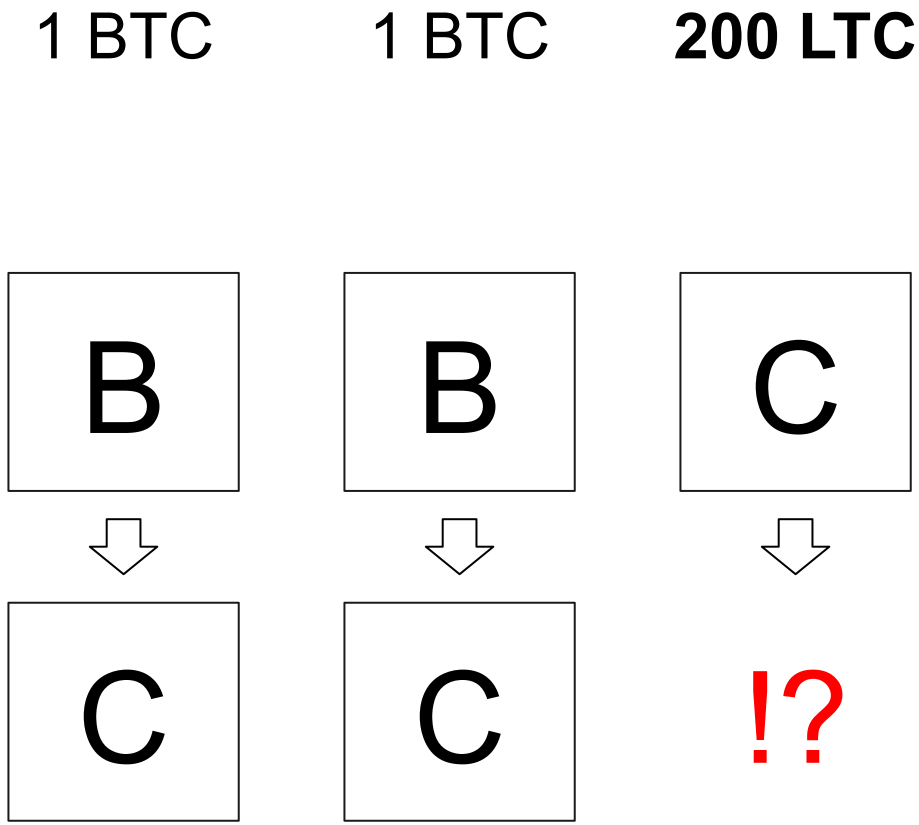
Swapping to smaller amounts



Possible with other coins



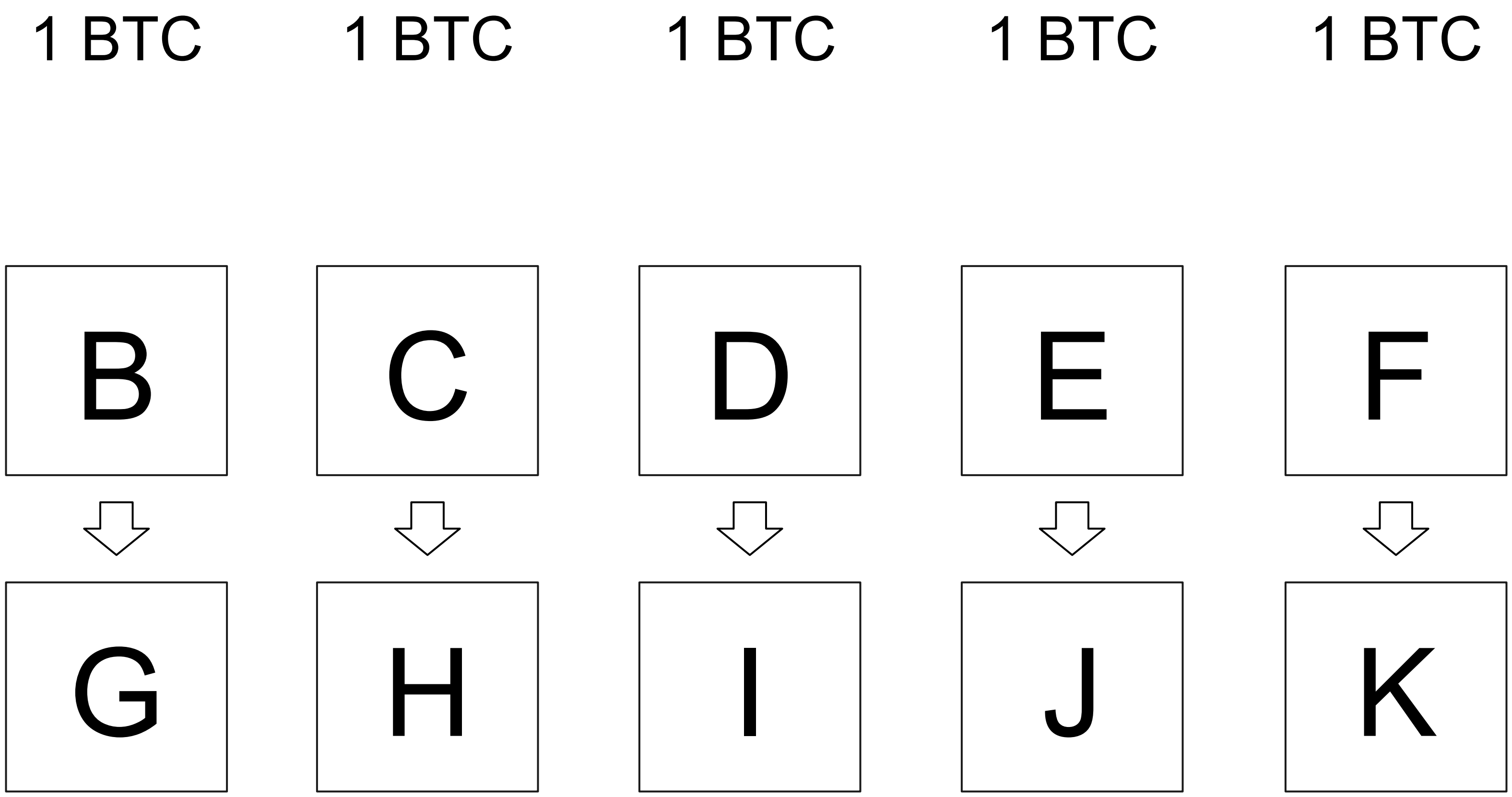
Money can get stolen if not atomic!



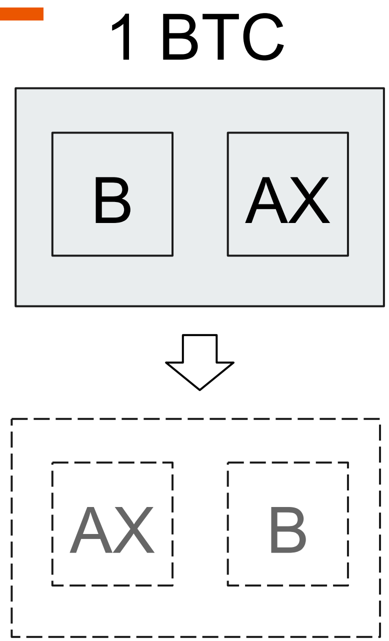
CoinSwap (off-chain coinjoin)

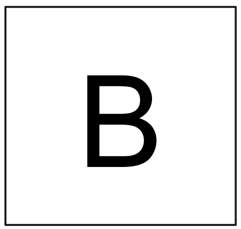
1 BTC	1 BTC	1 BTC	1 BTC	1 BTC
B	C	D	E	F

CoinSwap (off-chain coinjoin)

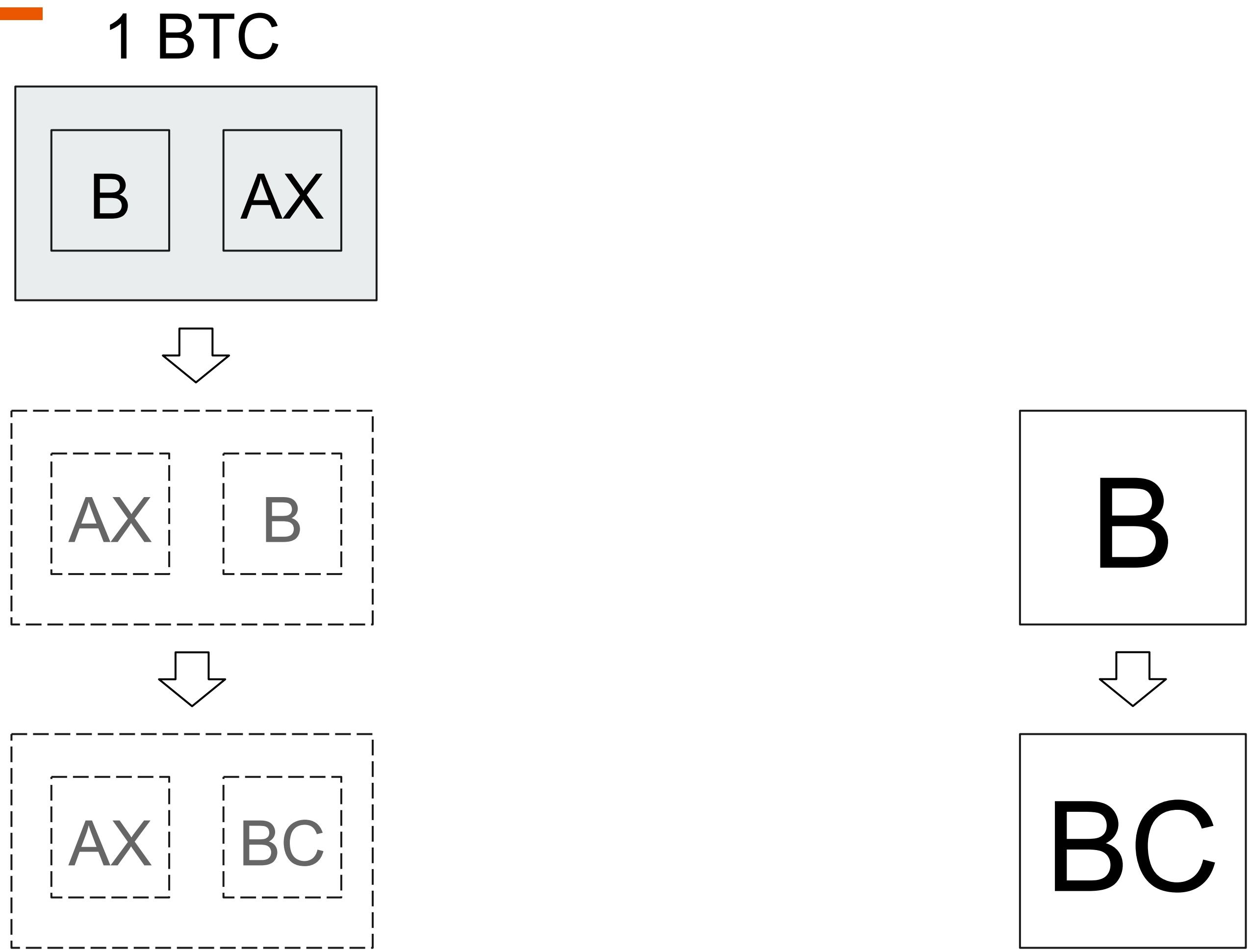


Lightning Channel Creation

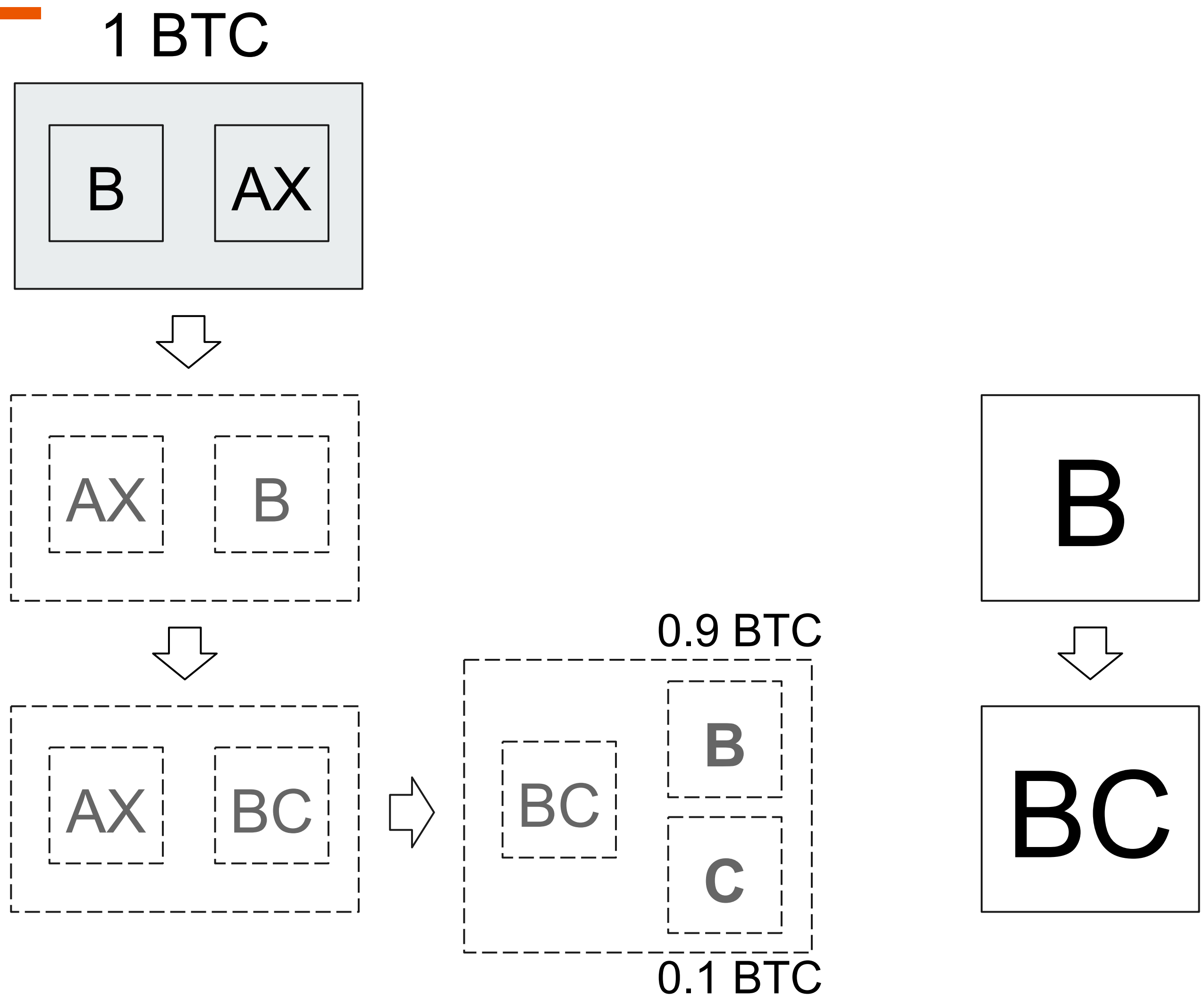




Lightning Channel Creation



Lightning Channel Creation



Schnorr Crash Course

Schnorr

- Promise: simple math

- A **solid** understanding of the basics makes it possible to understand many cool things:

  Taproot, Pedersen Commitments, Ring Signatures, Confidential Transactions, Mimblewimble, Bulletproofs*, Adaptor Sigs...

- Don’t just understand it, grok it!

One Basic Assumption

- Cryptography uses **special numbers** (curve points)

- These **special numbers** are limited:
  you can add (+) and subtract (-), *nothing* else

- Example: **5** + **3** = **8**   **5** * **3** = ??

Capital Letters

- **Special numbers** are written in capital letters

- Example: $A + B = C$

- We *can* multiply **special numbers** by normal numbers:
  $2A = A + A$ $\qquad$ $3A = A + A + A$

- We are still only using addition!

Possible to calculate?

A + B	Yes, we can add two **special numbers**
2A + 2A	Yes, this is A + A + A + A = 4A
2C + 3C	Yes, this is 5C
2A - 3B	Yes, (A + A) - (B + B + B)
B * B	No, we can only add/subtract **special numbers**
A * 2C	No, we can only add/subtract **special numbers**
2D / 3D	No, we can only add/subtract **special numbers**

Possible to calculate **x** and **y**?

2E + **x**E = 5E	Yes, x = 3 ((E + E) + (E + E + E))
xF + **y**F = 8F	Infinite possibilities (e.g. x=108, y=-100)
6G + **x**G = **y**G	Infinite possibilities (e.g. x=94, y=100)

You can’t resolve two variables

Reversing a calculation

- If 5A = E, can we get **x**=5 from knowing just **x**A = E?
- Trial and error:

  E - A = D

  D - A = C

  C - A = B

  B - A = A   Found it!
- Can we reverse 97639273952850352803528532A = F?
  Takes forever... Impossible!

Efficiently going forward

- Isn't 97639273952850352803528532A = F equally slow to calculate? No, because:

  A + A = 2A
  2A + 2A = 4A
  4A + 4A = 8A (and so on)

- Doubling the number with each step makes it quick to get to a huge number (but impossibly slow to reverse!)

Keys and Signatures

Private and Public Keys

- Given: starting point “G” (everybody knows G)
- We pick a huge random number as our **private key**:
  a = 97639273952850352803528532
- **private key** * G = **public key** (pseudonymous identity)
- **a**G = **A**

Proving you know the private key of A

- Note: this method has a flaw!
- Pick another huge random number r*G = R
- Calculate r + **a** = **s**
- Give R and **s** to the verifier
- Verifier calculates R + **A** = **s***G

Proving you know the private key of A

- Why does R + **A** = **s***G prove you know a?
- Recall our example: 6G + **x**G = **y**G two variables
- Calculating **s** requires knowledge of both secrets (r + **a**)
- Flaw: if R = r*G - **A**, then you're calculating R - **A** + **A**

Fixing the flaw and adding a message

- Introduce **e** = hash(R)
- Prover: r + **e***a** = **s**
- Verifier: R + **e***A** = **s***G

- Impossible to cheat:
  R = r*G - **e***A** (impossible: e depends on R (e.g. x = x - 2))

  Easy to add a message:
  e = hash(R, message)

Adaptor Signatures

Adaptor Signatures

- High level: incomplete signatures, which can be completed with a secret from another signature

- Normal Schnorr: $R + \mathbf{e}*\mathbf{A} = \mathbf{s}*G$
- Incomplete adaptor signature: $(R+\mathbf{D}) + \mathbf{e}*\mathbf{A} = \mathbf{s}*G$
- Completed adaptor signature: $(R+\mathbf{D}) + \mathbf{e}*\mathbf{A} = (\mathbf{s}+\mathbf{d})*G$

- Multiple secrets can be combined for multiple sigs:
  D1 + **D2** + **D3** = **D** (MuSig)

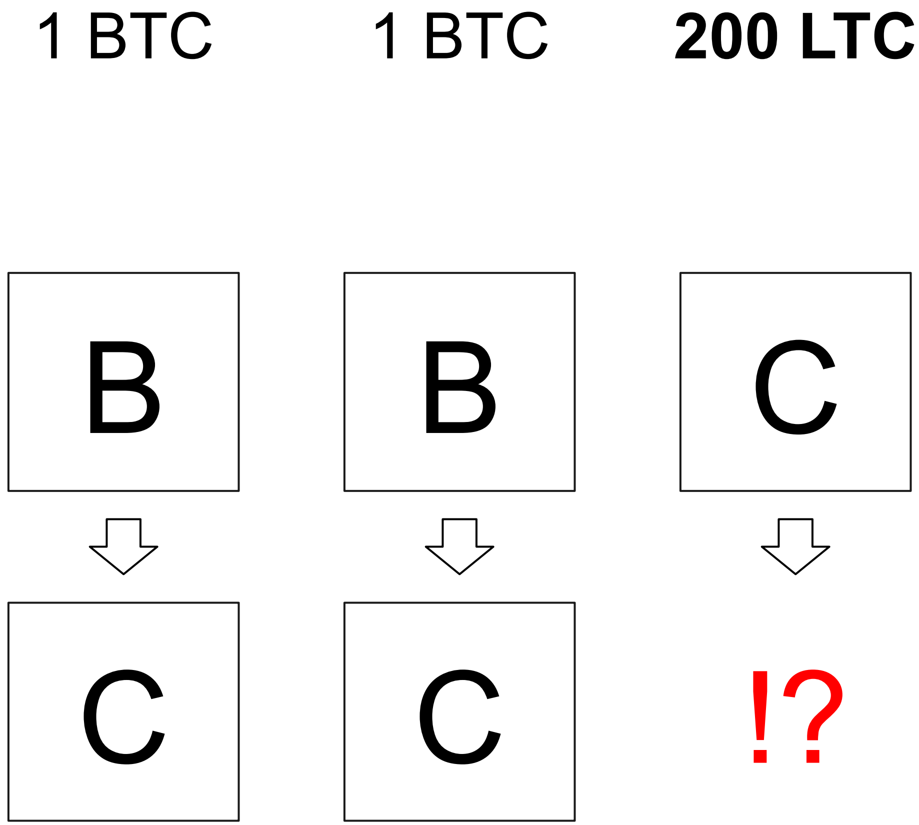
Adaptor Signatures

- Three incomplete adaptor sigs, everyone gets a copy:
  (R1+**D**) + **e***__A__ = **s1**   *G
  (R2+**D**) + **e***__B__ = **s2**   *G
  (R3+**D**) + **e***__C__ = **s3**   *G

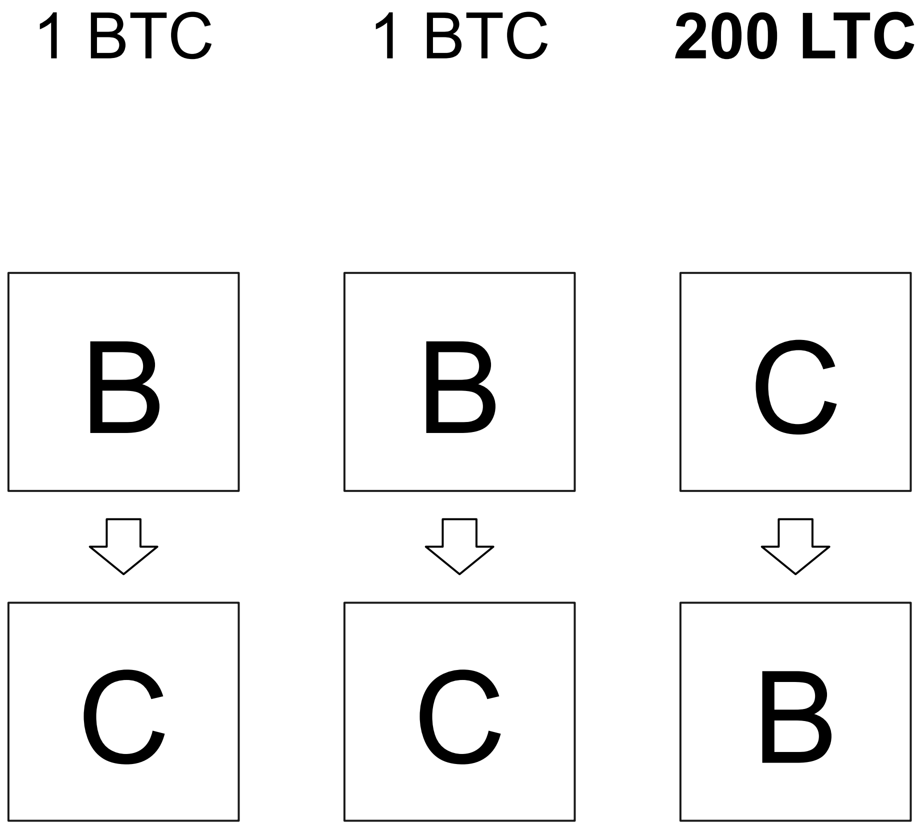
- Everyone shares their secrets: **d1** +**d2** + **d3** = **d**

- Can't withhold a secret, publishing your sig reveals **d**:
  e.g. [**s**, R] where **s** = **s3** + **d**, meaning **s** - **s3** = **d**

Recall our atomic issue



Now B can complete the signature



Thank you