Cashu

A Chaumian ecash protocol for Bitcoin

calle@lnbits.com

@callebtc

Blinding custody

Two main risks of custodial Bitcoin services:

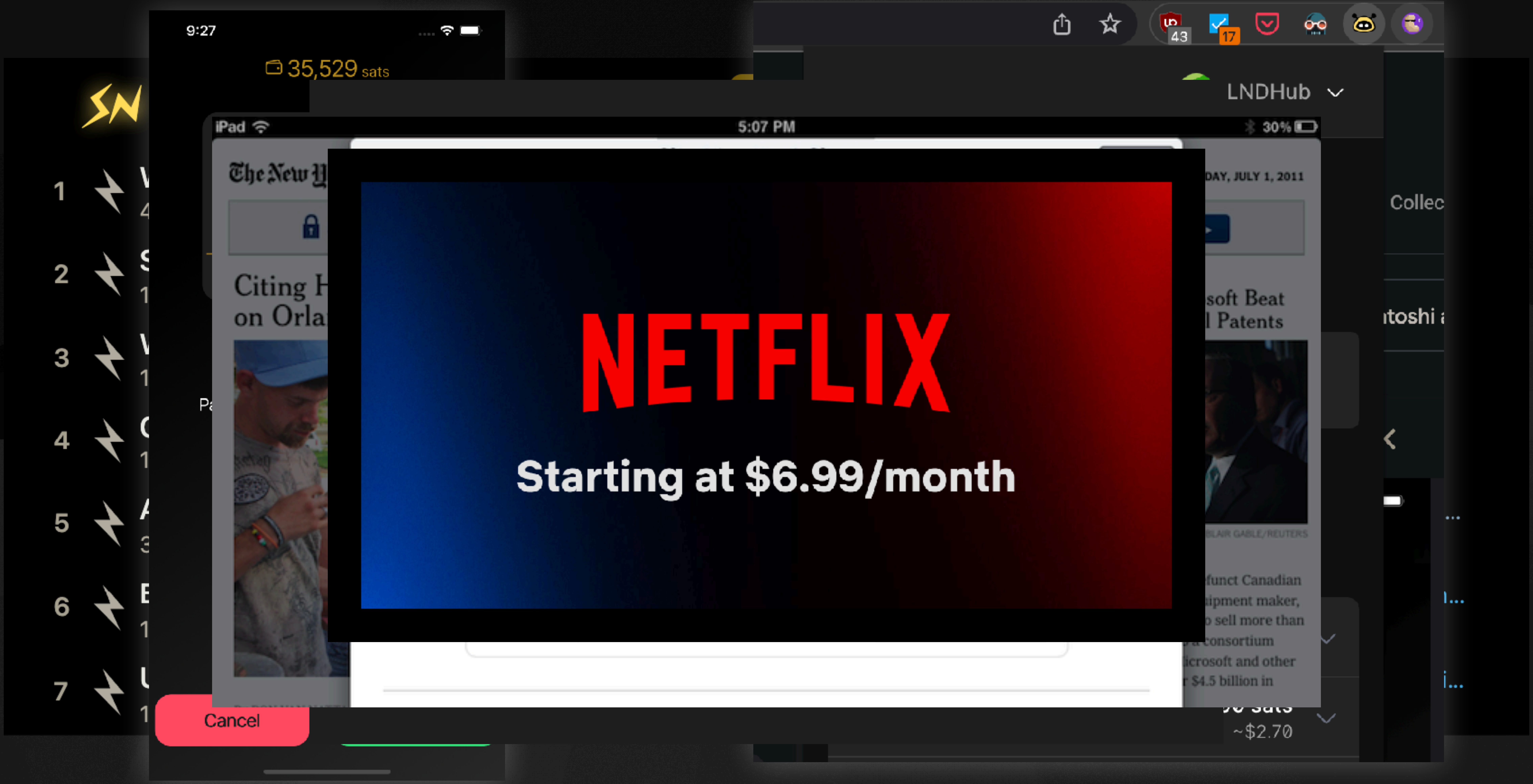
1) Custodial risk
2) No privacy

Problem: A custodian **must** know your transaction history, balance, payments in and out of the system.

Solution: An open and interoperable Chaumian Ecash system.

We use custodians everywhere.

They infringe on our privacy especially with small and frequent payments.


9:27
35,529 sats
LNDHub
iPad
5:07 PM
30%
NETFLIX
Starting at $6.99/month
Cancel
~$2.70

Classical custodianship

I'm Alice (id=28bd2f...)
Send 1 BTC to Carol (id=28bd2f...)
Does Alice have a balance of ≥ 1 BTC?
My balance updated!
Alice
Bob
Carol
Dave -0.5 BTC
Alice +1 BTC
Alice -1 BTC
Carol +1 BTC
Ledger

Requires Alice and Bob have an **account**

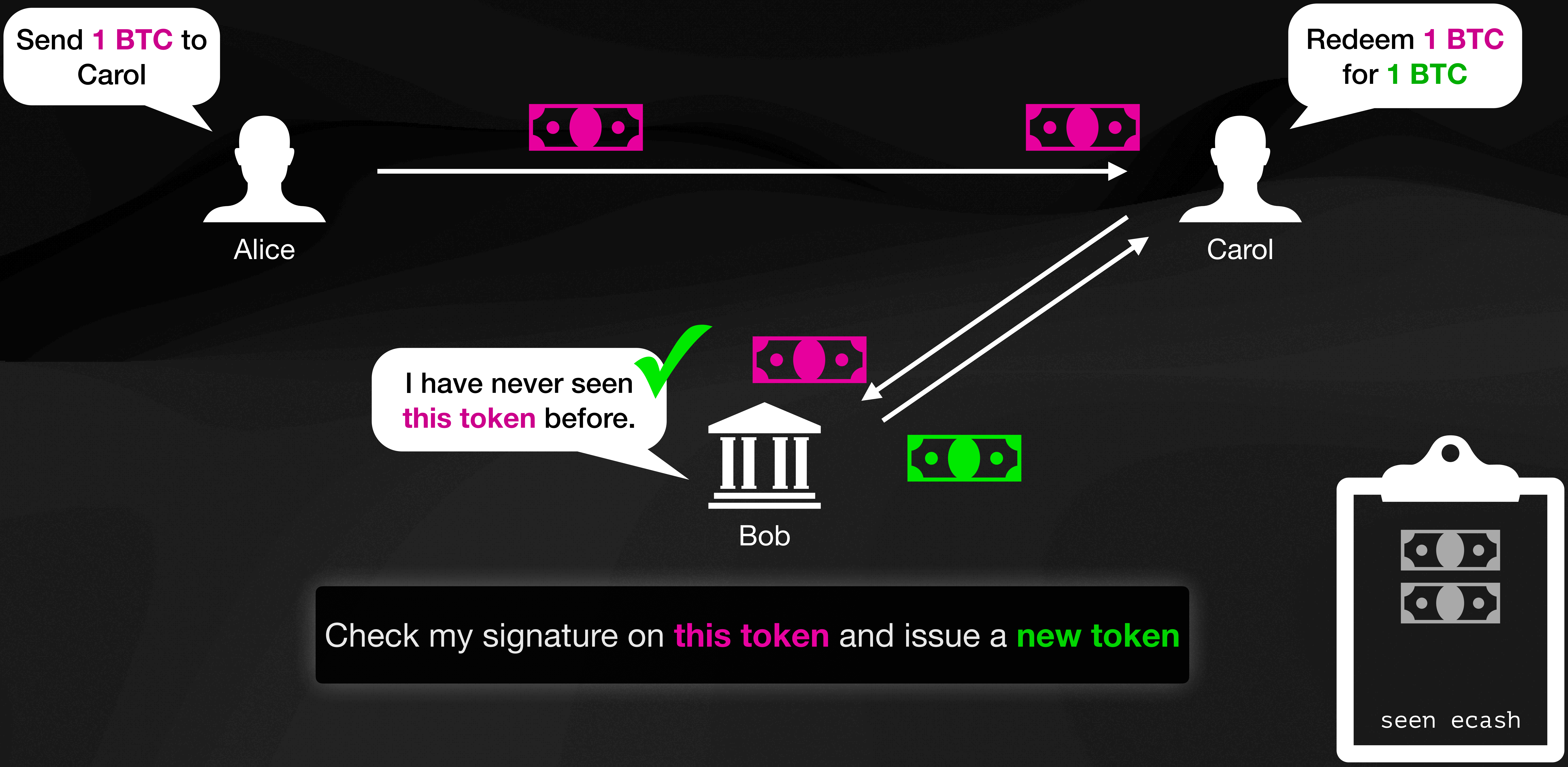
The custodian MUST know

your user ID, your balance, and your transaction history

to function.

Chaumian ecash

Chaumian ecash



Check my signature on this token and issue a new token

Properties of Ecash

UNTRACEABLE

The mint does know very little about the financial activity of its users.

BEARER TOKEN

The data *is* the money. Ecash can be embedded in data packages.

PUSH UX

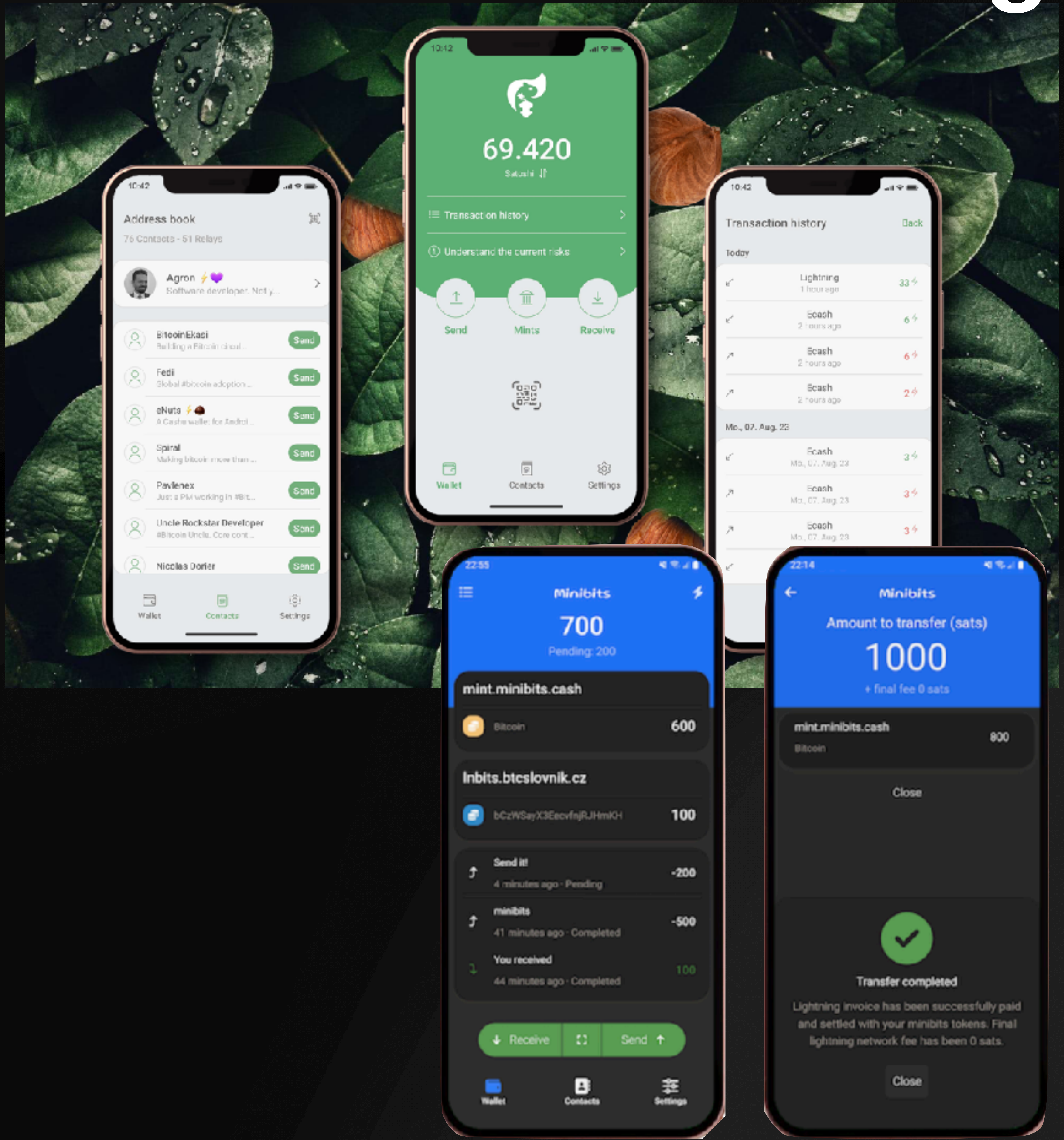
Payer "throws" money at receiver. With an online inbox, can receive while offline.

PROGRAMMABLE

Complex spending conditions for ecash enforced by the mint.

Cashu

Cashu Lightning wallets



eNuts.cash
(iOS Testflight & Android)

minibits.cash
(Android)

nutstash.app
(PWA)

cashu.me
(PWA)

Implementations

Mints	Wallets	Integrations	New use cases
Nutshell	Nutshell	Snort	ProxNut
LNbits	Feni	Amethyst	X-Cashu
cashu-rs	Nutstash	Redeem	Katzenpost
Moksha	Cashu.me	Spacenut	Nutminer
	eNuts		
	Cashcrab		
	Moksha		
	Minibits		

More info: **https://docs.cashu.space**

Milestones

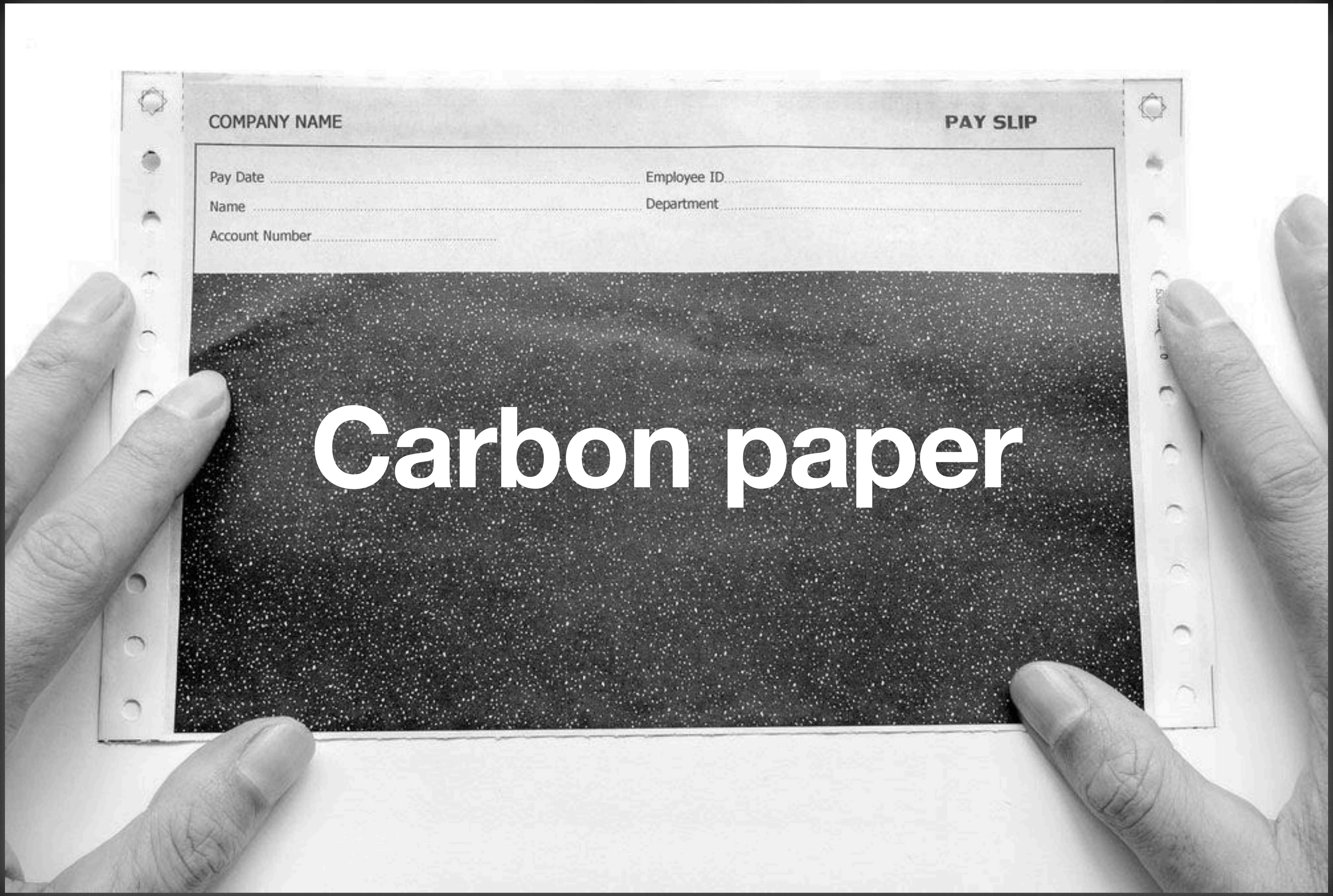
Since Q3 2022

- ✓ **Bitcoin Lightning** integration
- ✓ **Deterministic ecash** derivation and seed phrase backups
- ✓ **Programmable ecash** with complex spending conditions (P2PK, multisig)
- ✓ **Proof-of-Liabilities** scheme for public auditability of ecash mints
- ✓ **Receiver-offline** transactions that are verifiably final
- ✓ **Libraries** in Python, Rust, Golang, TypeScript
- ✓ **Mobile wallets** for iOS, Android, and PWA

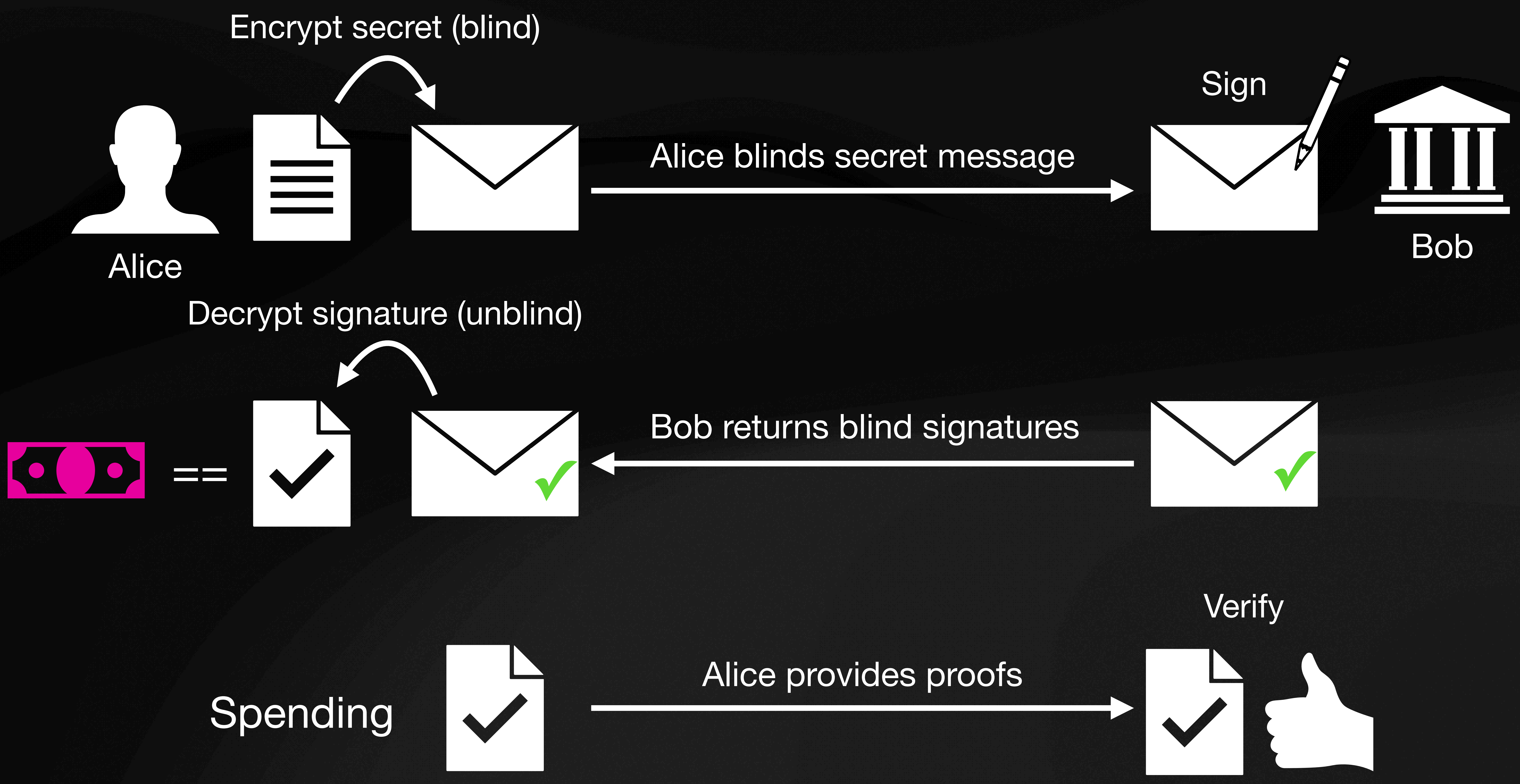
Blind signatures

Blind signatures

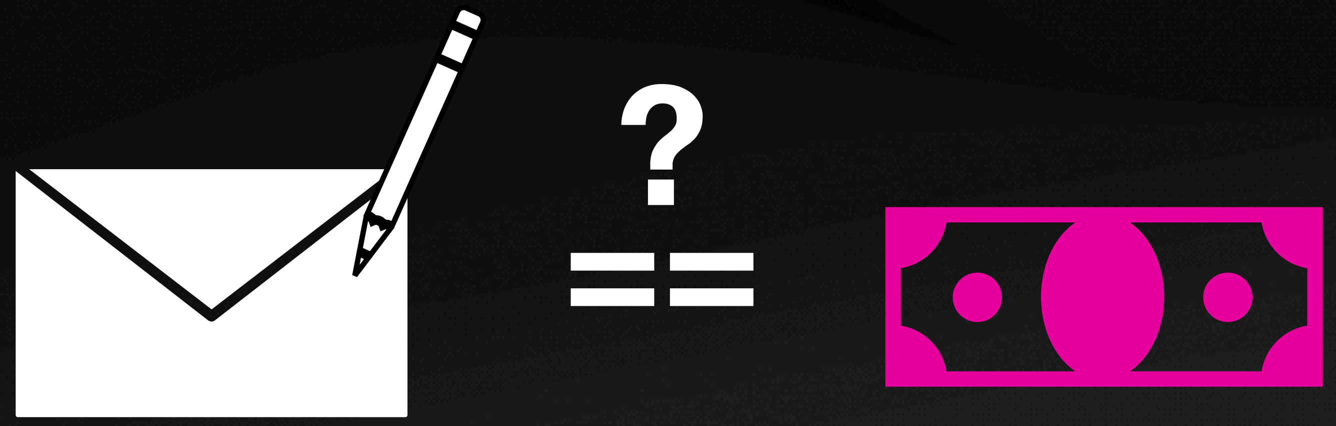
Blind signatures allow you to **sign** a message that you have never seen and to **verify** your signature once the message is revealed to you.



Blind signatures


Encrypt secret (blind)
Sign
Alice blinds secret message
Alice
Bob
Decrypt signature (unblind)
Bob returns blind signatures
==
Verify
Spending
Alice provides proofs

The signature is unlinked from the ecash token.

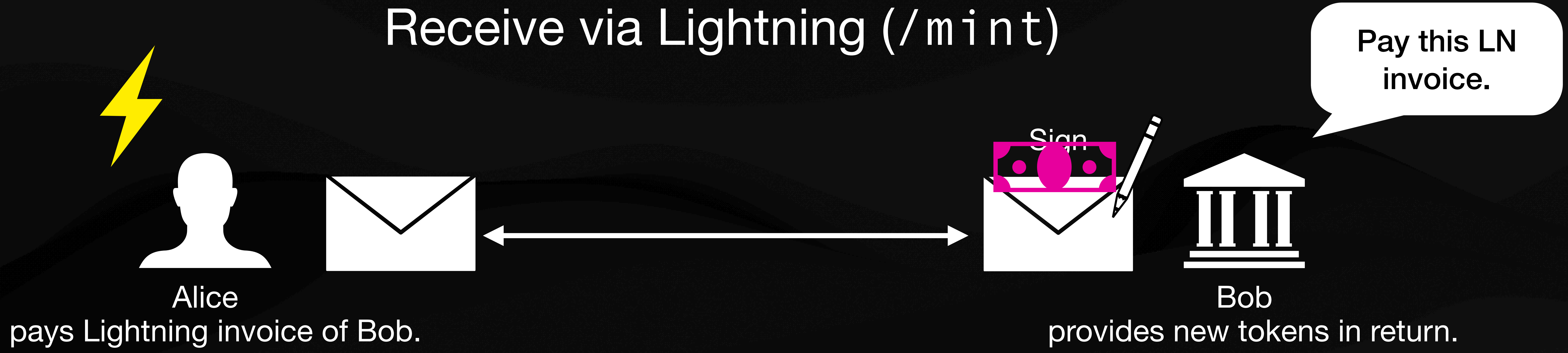


The mint does not know which ecash token it is signing.

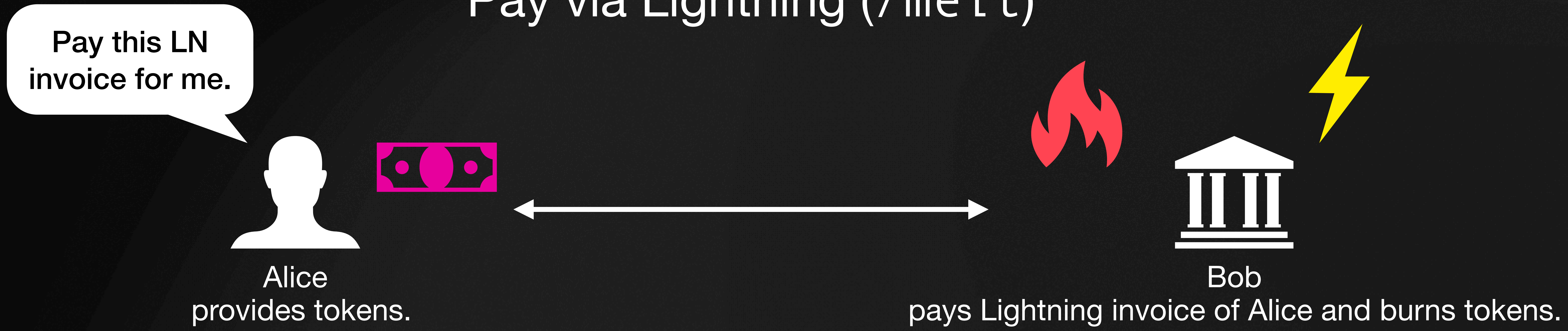
Cashu on Lightning

Cashu on Lightning

Receive via Lightning (`/mint`)

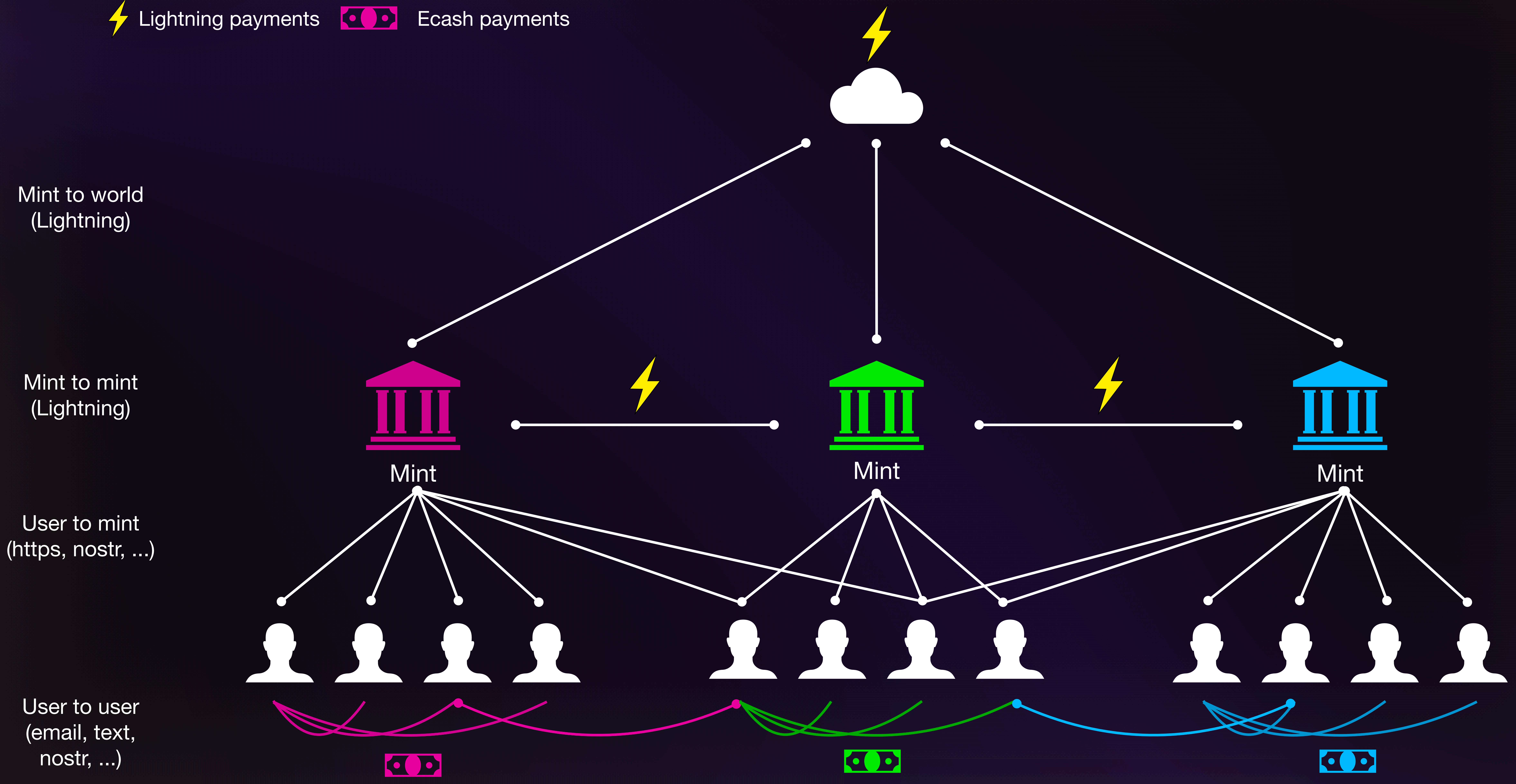

Pay this LN invoice.
Sign
Alice
pays Lightning invoice of Bob.
Bob
provides new tokens in return.

Pay via Lightning (`/melt`)


Pay this LN invoice for me.
Alice
provides tokens.
Bob
pays Lightning invoice of Alice and burns tokens.

Lightning is the connecting tissue

⚡ Lightning payments 💵 Ecash payments



Programmable ecash

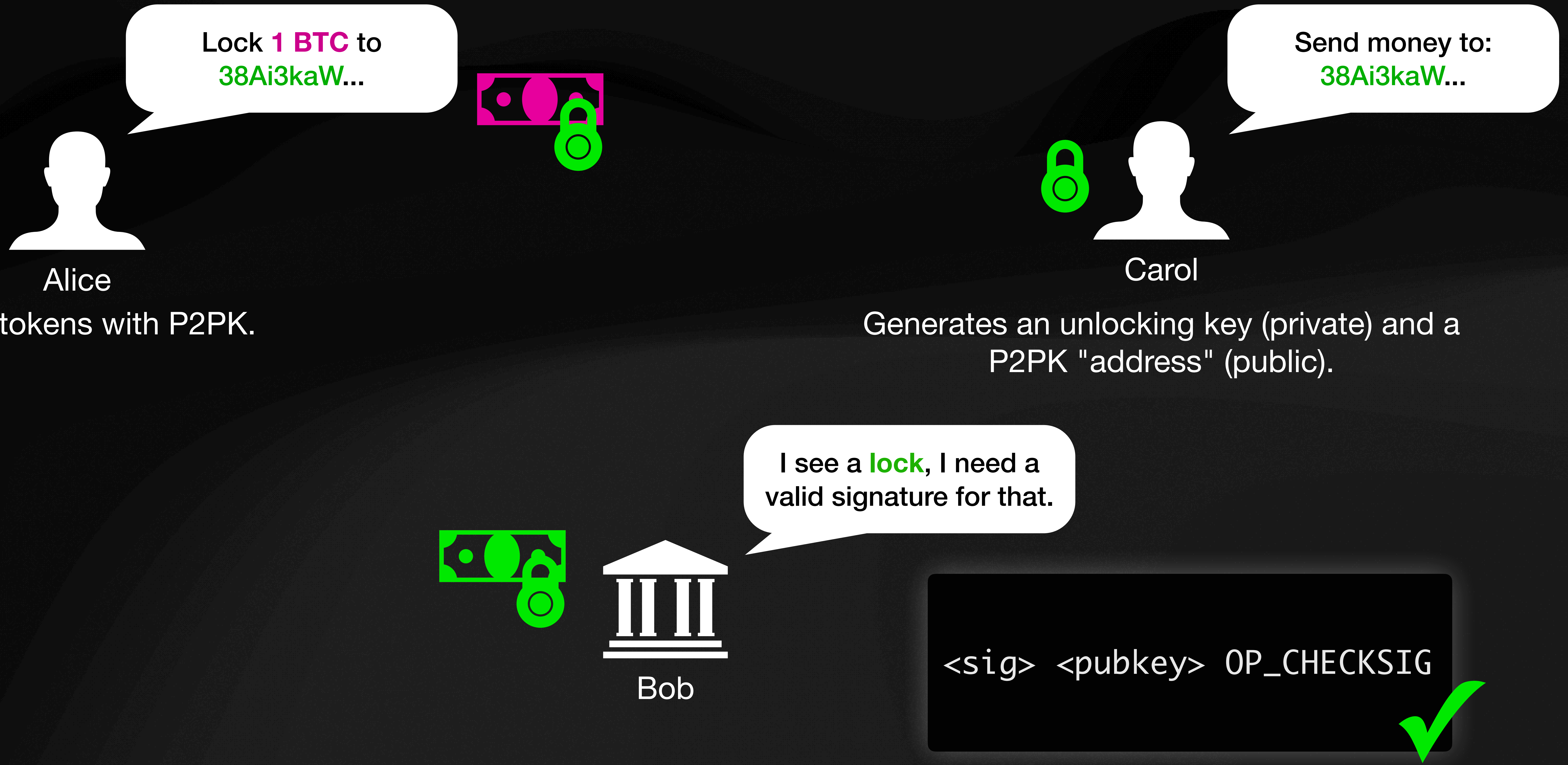
Programmable ecash

We can attach spending conditions to ecash. Spending conditions are enforced by the mint.

Like Bitcoin UTXOs:
To spend locked ecash, users must provide a valid unlocking witness.

Example: Pay to Public Key (P2PK).

Pay to public key (P2PK)


Lock 1 BTC to 38Ai3kaW...
Send money to: 38Ai3kaW...
I see a lock, I need a valid signature for that.

Alice
Locks tokens with P2PK.

Carol
Generates an unlocking key (private) and a P2PK "address" (public).

Bob

```
<sig> <pubkey> OP_CHECKSIG
```

✓

Pay to public key (P2PK)

Post ecash publicly

Example: Zap nostr posts with ecash.

Receiver can remain offline

"*I see the ecash locked to me, that's enough*."

Enables (very) high-frequency payments

Receiver can defer round trips to mint to the future.

Hash timelock contracts (HTLC)

Atomic ecash swaps

Example: Exchange ecash between mints.

Lightning submarine swaps

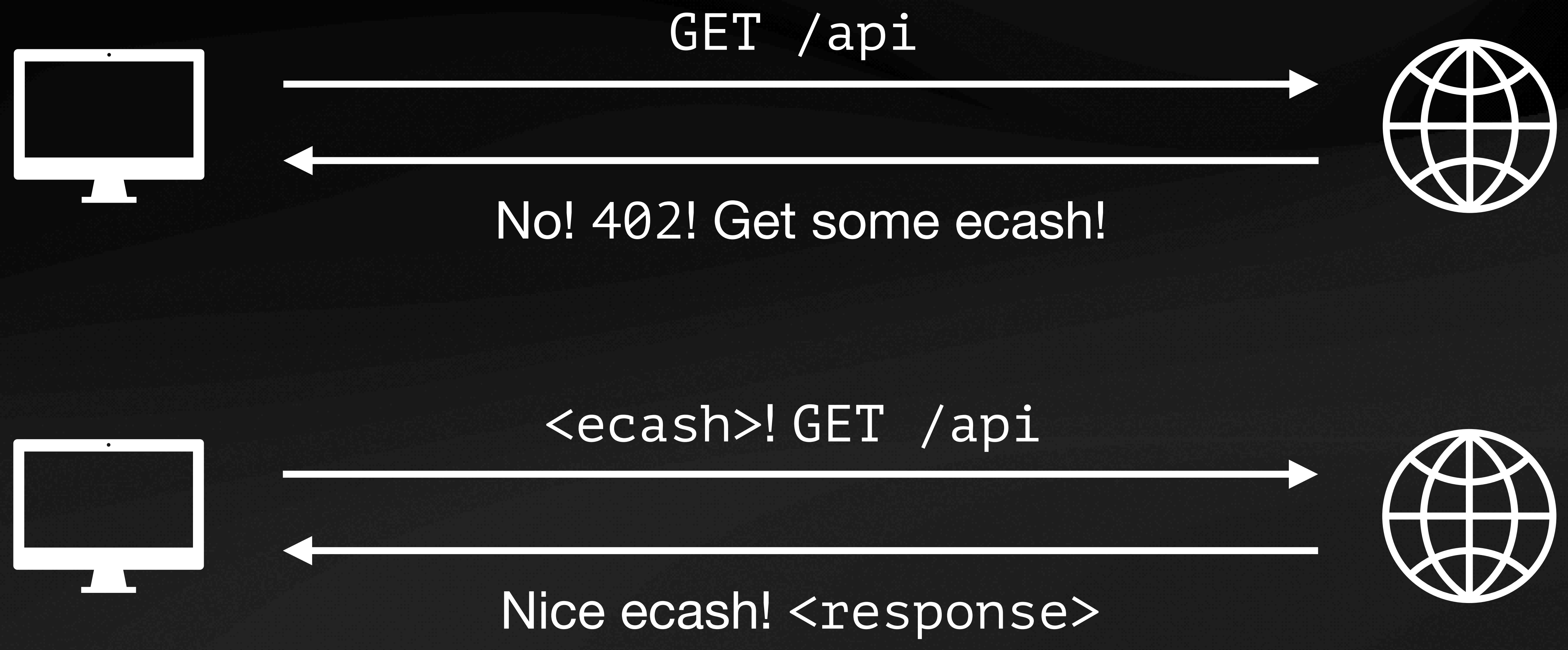
Atomically swap ecash for a successful Lightning payment

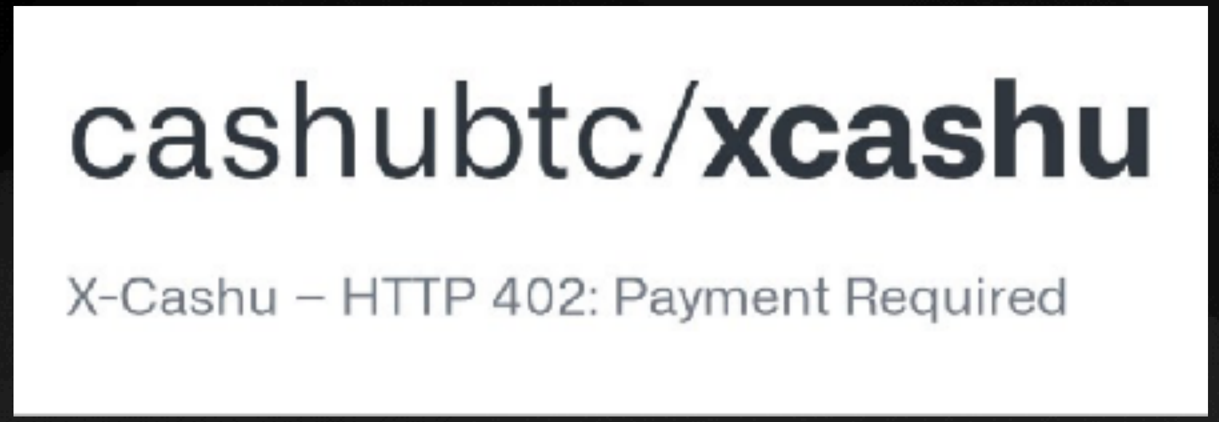
Route Lightning payments?

Lightning HTLC routes can "shortcut" through an ecash system

X-Cashu

HTTP 402: Payment required




cashubtc/**xcashu**

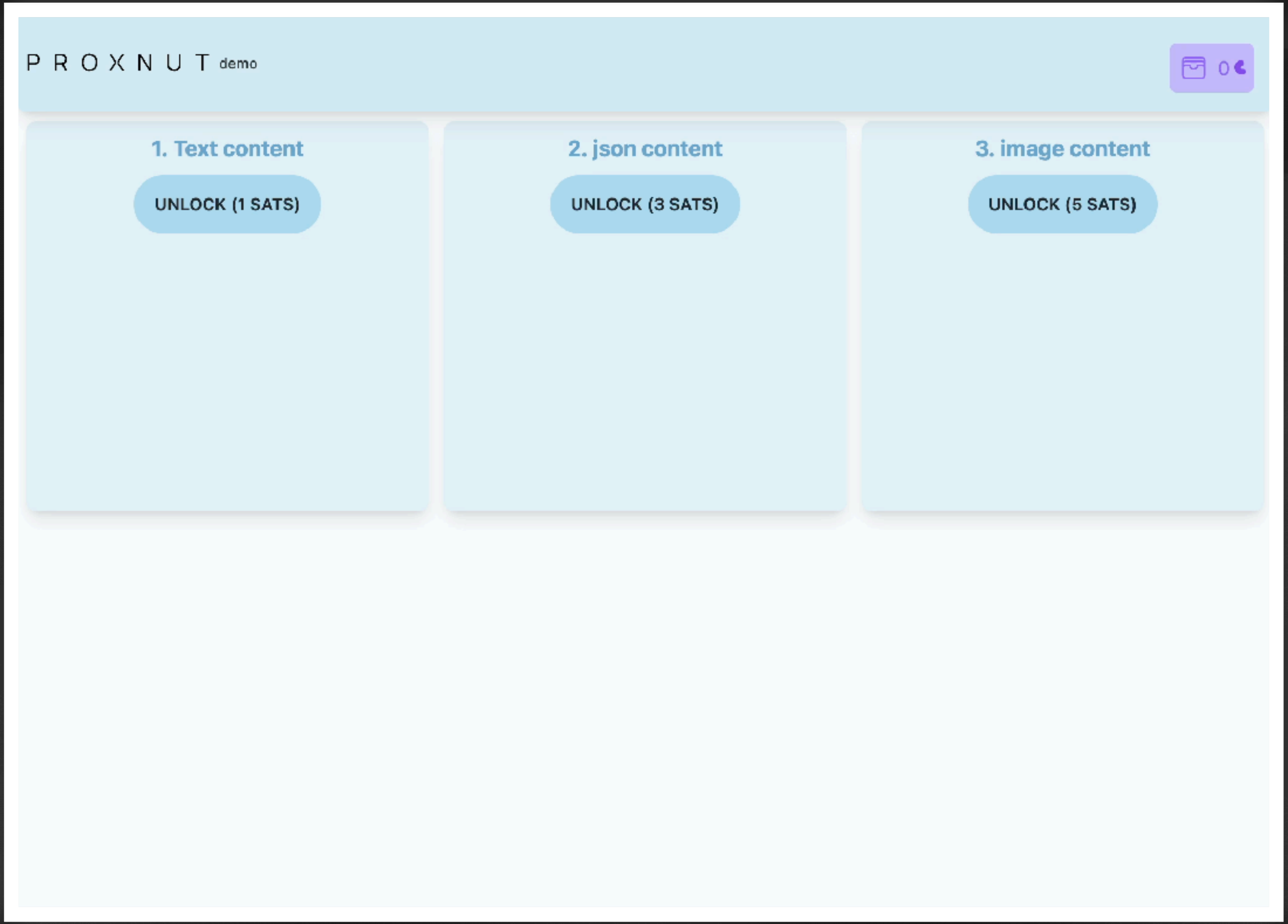
X-Cashu – HTTP 402: Payment Required

User attaches ecash directly to request

Server **can not know** which user paid how much.

PROXNUT

Web widget for privacy-preserving paywalls with Cashu



Onion-routed Ecash

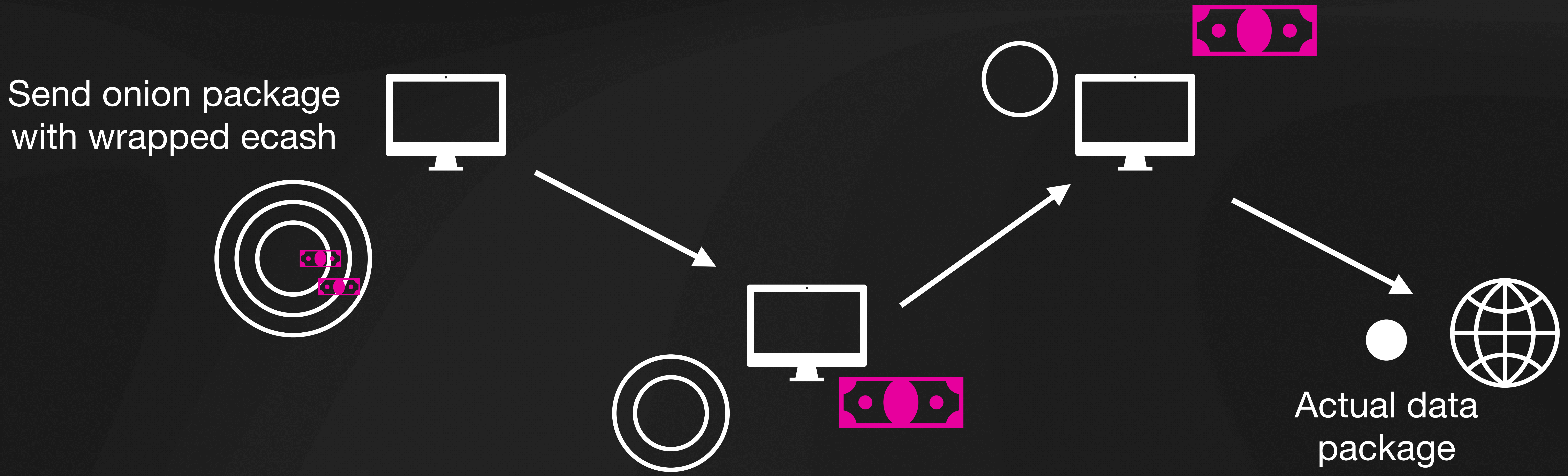
Use Tor or Katzenpost privacy services for a fee

Problem: Which payment method is fast and private enough?

Idea: Add ecash payment inside the request itself.

Idea²: Wrap multiple payments in layers of an onion.

WIP integration in Katzenpost (mixnet)



We're looking for contributors

Python, Rust, TypeScript, Golang
UX Design, Documentation, Community

ecashhackday.github.io

https://cashu.space

Try Nutstash wallet

https://nutstash.app/

May the nut be with you 🧡

@callebtc