GNU

Funded by the
European Union

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
State Secretariat for Education,
Research and Innovation SERI

taler.net
taler@twitter

Christian Grothoff
grothoff@taler.net

Agenda

- Motivation & Background
- GNU Taler: Introduction
- Component Zoo
- Protocol Basics
- Attacks & Defenses
- Offline payments
- Programmable money: Age restrictions
- Software development & deployment
- Performance
- Blockchain integration: Project Depolymerization
- Future Work & Conclusion

TALER

Surveilance concerns

- Everybody knows about Internet surveilance.
- But is it **that** bad?

Surveilance concerns

- Everybody knows about Internet surveilance.
- But is it **that** bad?
  - You can choose when and where to use the Internet
  - You can anonymously access the Web using Tor
  - You can find open access points that do not require authentication
  - IP packets do not include your precise location or name
  - ISPs typically store this meta data for days, weeks or months

A Social Problem

This was a question posed to RAND researchers in 1971:

> *"Suppose you were an advisor to the head of the KGB, the Soviet Secret Police. Suppose you are given the assignment of designing a system for the surveillance of all citizens and visitors within the boundaries of the USSR. The system is not to be too obtrusive or obvious. What would be your decision?"*

A Social Problem

This was a question posed to RAND researchers in 1971:

> *"Suppose you were an advisor to the head of the KGB, the Soviet Secret Police. Suppose you are given the assignment of designing a system for the surveillance of all citizens and visitors within the boundaries of the USSR. The system is not to be too obtrusive or obvious. What would be your decision?"*

"I think one of the big things that we need to do, is we need to get away from true-name payments on the Internet. The credit card payment system is one of the worst things that happened for the user, in terms of being able to divorce their access from their identity." –Edward Snowden, IETF 93 (2015)

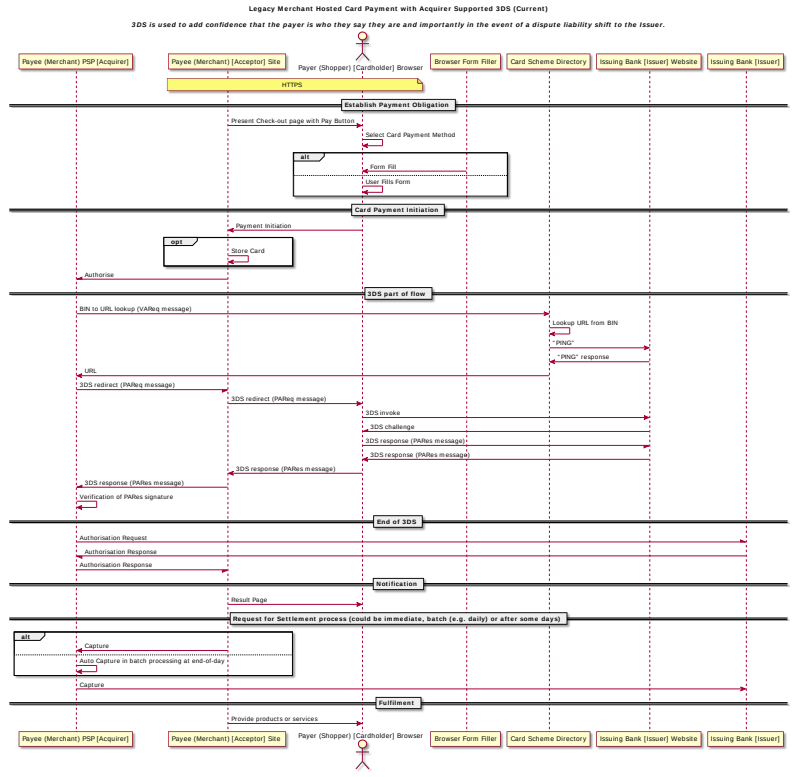
What is worse:

- When you pay by CC, the information includes your name
- When you pay in person with CC, your location is also known
- You often have no alternative payment methods available
- You hardly ever can use someone else's CC
- Anonymous prepaid cards are difficult to get and expensive
- Payment information is typically stored for at least 6 years

Banks have Problems, too!

3D secure (“verified by visa”) is a nightmare:

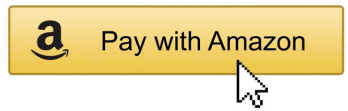
- Complicated process
- Shifts liability to consumer
- Significant latency
- Can refuse valid requests
- Legal vendors excluded
- No privacy for buyers



Online credit card payments will be replaced, but with what?

The Bank's Problem

- Global tech companies push oligopolies
- Privacy and federated finance are at risk
- Economic sovereignty is in danger



TALER

Predicting the Future

- Google and Apple will be your bank and run your payment system
- They can target advertising based on your purchase history, location and your ability to pay
- They will provide more usable, faster and broadly available payment solutions; our federated banking system will be history
- After they dominate the payment sector, they will start to charge fees befitting their oligopoly size
- Competitors and vendors not aligning with their corporate "values" will be excluded by policy and go bankrupt
- The imperium will have another major tool for its financial warfare

The Distraction: Bitcoin

- Unregulated payment system and currency:

  $\Rightarrow$ lack of regulation is a feature!
- Implemented in free software
- Decentralised peer-to-peer system

The Distraction: Bitcoin

- Unregulated payment system and currency:

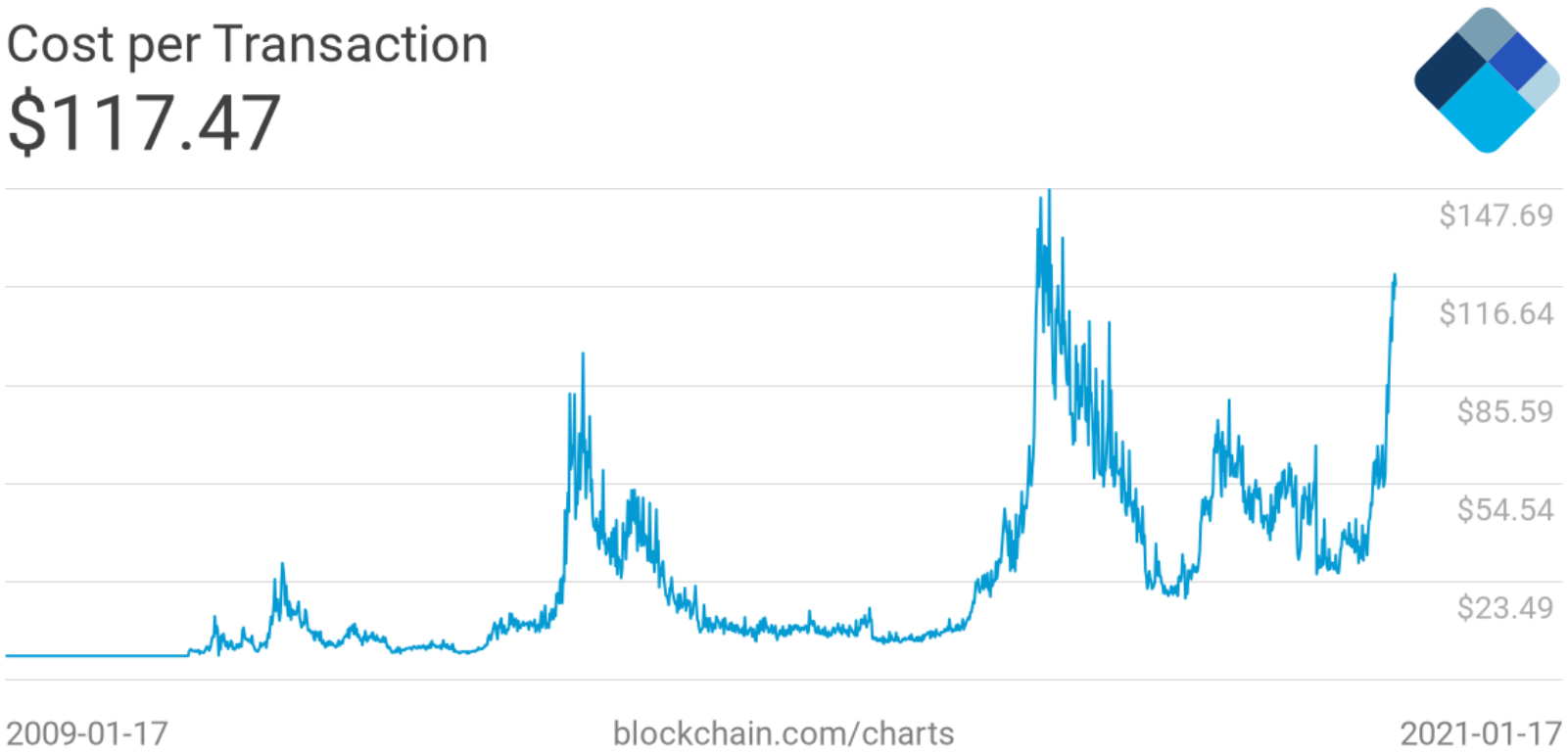
  $\Rightarrow$ lack of regulation is a feature!
- Implemented in free software
- Decentralised peer-to-peer system
- Decentralised banking requires solving Byzantine consensus
- Creative solution: tie initial accumulation to solving consensus

The Distraction: Bitcoin

- Unregulated payment system and currency:
  $\Rightarrow$ lack of regulation is a feature!
- Implemented in free software
- Decentralised peer-to-peer system
- Decentralised banking requires solving Byzantine consensus
- Creative solution: tie initial accumulation to solving consensus
  $\Rightarrow$ Proof-of-work advances ledger
  $\Rightarrow$ Very expensive banking

bitcoin?

Background: https://blockchain.com/charts/



Current average transaction value: $\approx$ 1000 USD

TALER

Zerocoin?

Bitcoin cryptography is rather primitive:

All Bitcoin transactions are public and linkable!

$\Rightarrow$ no privacy guarantees

$\Rightarrow$ enhanced with "laundering" services

ZeroCoin, CryptoNote (Monero) and ZeroCash (ZCash) offer anonymity.

Do you want to have a libertarian economy?

Do you want to live under total surveillance?

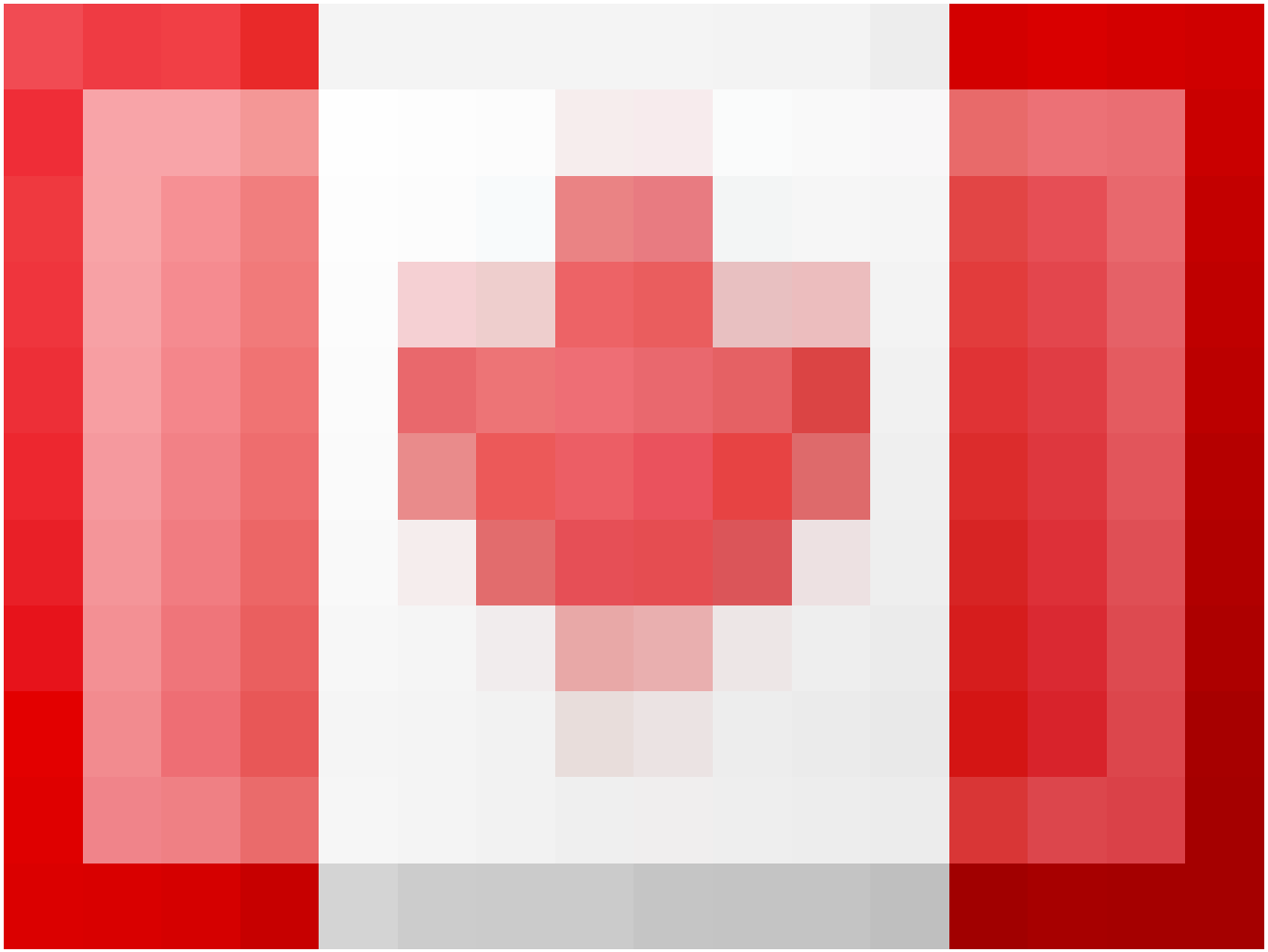
TALER

The Bank of International Settlements

Central Bank Digital Currency vs. Cash

TALER

The Emergency Act of Canada[1]



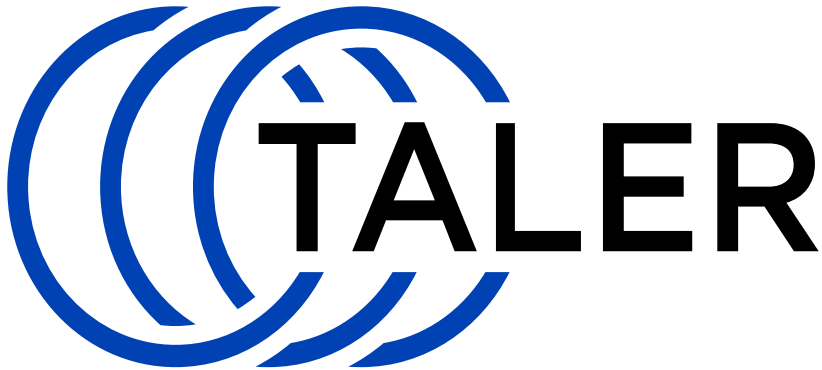
https://www.youtube.com/watch?v=NehMAj492SA (2'2022)

[1] Speech by Premier Kenney, Alberta, February 2022

GNU Taler: Introduction

TALER

Digital cash, made **socially responsible**.



Privacy-Preserving, Practical, Taxable, Free Software, Efficient

What is Taler?

https://taler.net/en/features.html

Taler is

- a Free/Libre software *payment system* infrastructure project
- ... with a surrounding software ecosystem
- ... and a company (Taler Systems S.A.) and community that wants to deploy it as widely as possible.

However, Taler is

- *not* a currency or speculative asset
- *not* a long-term store of value
- *not* a network or instance of a system
- *not* decentralized
- *not* based on proof-of-work or proof-of-stake

Design principles

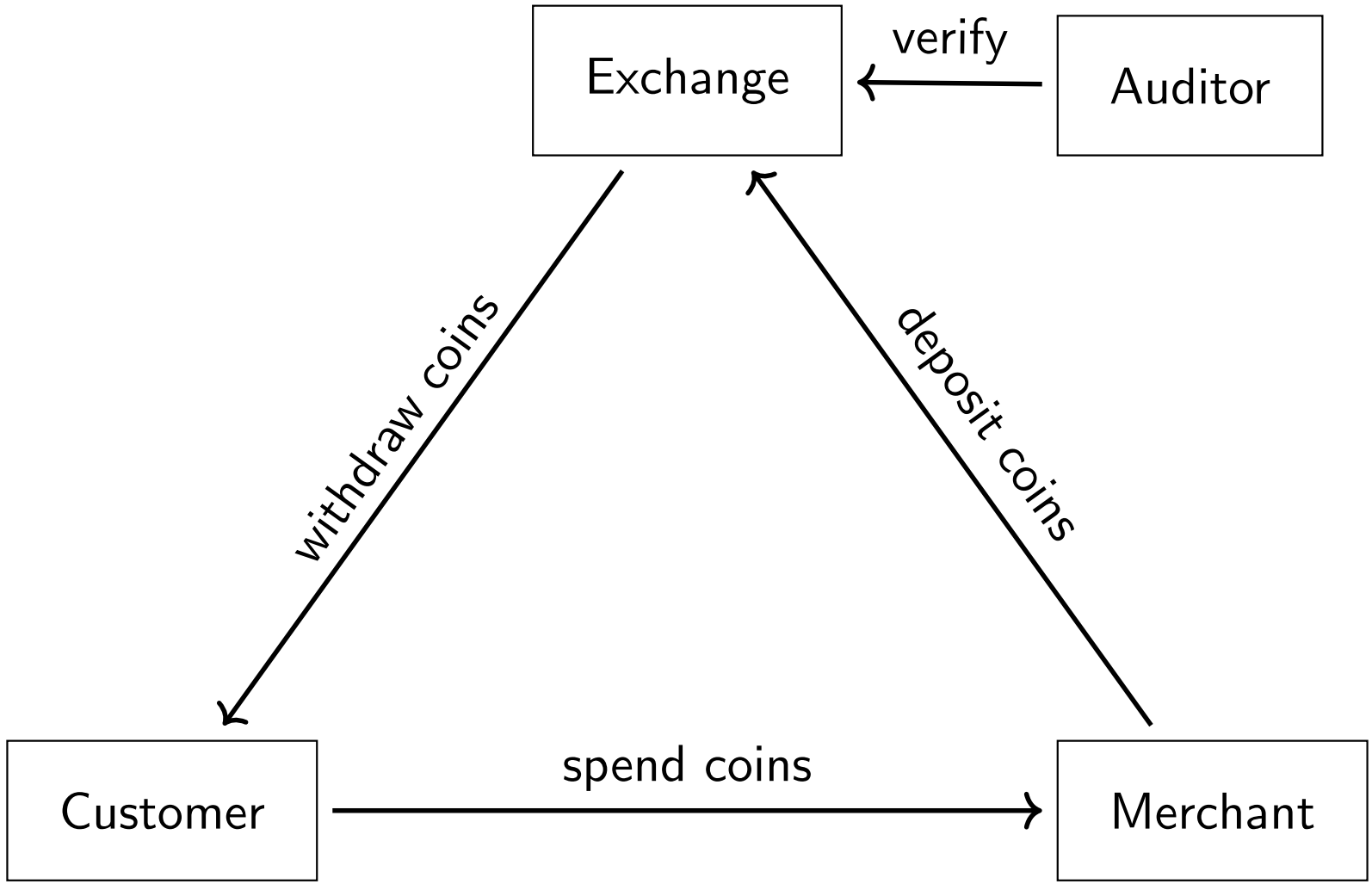
https://taler.net/en/principles.html

GNU Taler must ...

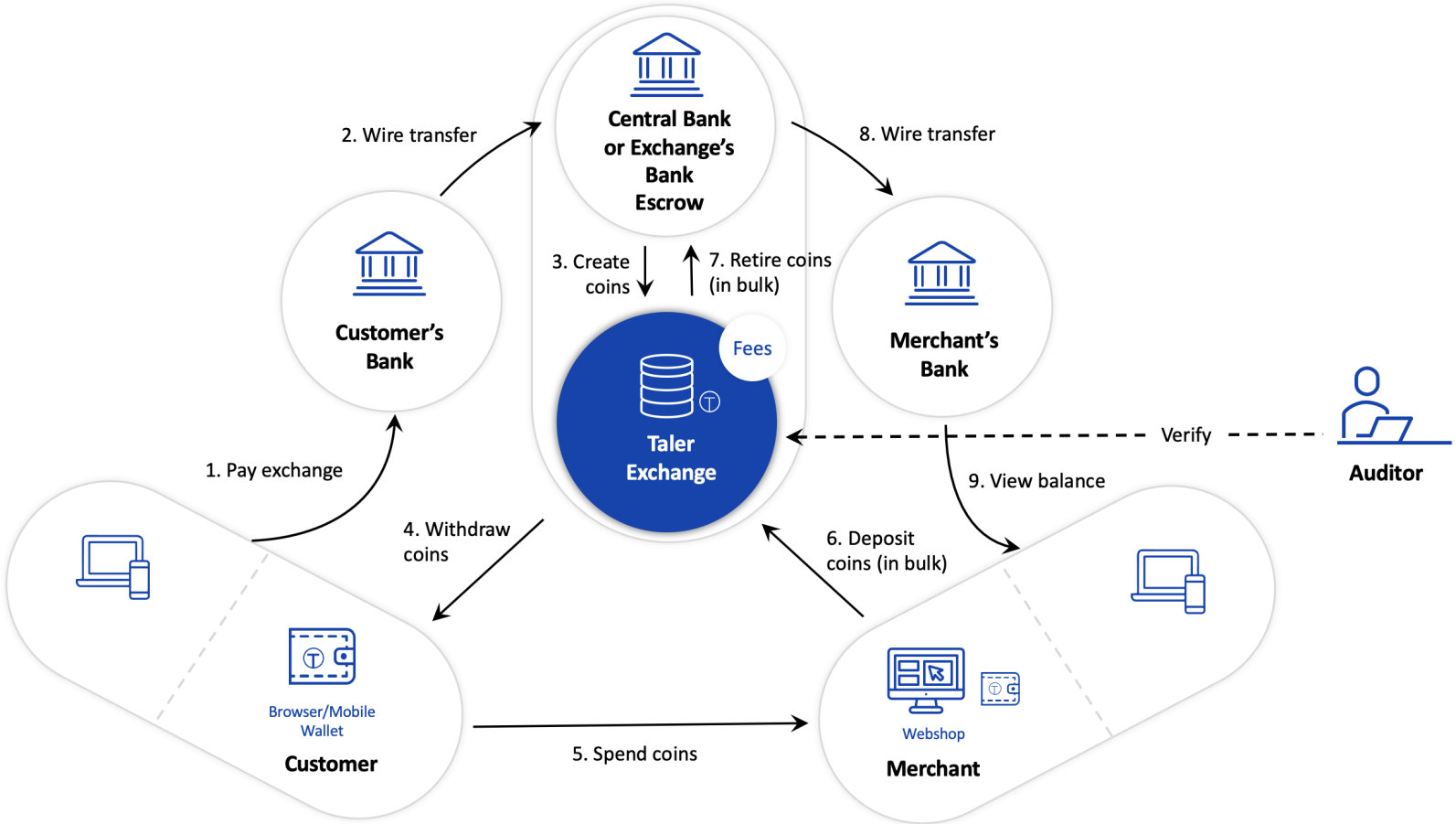
1. ... be implemented as **free software**.
2. ... protect the **privacy of buyers**.
3. ... enable the state to **tax income** and crack down on illegal business activities.
4. ... prevent payment fraud.
5. ... only **disclose the minimal amount of information necessary**.
6. ... be usable.
7. ... be efficient.
8. ... avoid single points of failure.
9. ... foster **competition**.

TALER

Taler Overview



Architecture of Taler



Consumer Impact of Taler

- **Convenient:** pay with one click instantly — in Euro, Dollar, Yen or Bitcoin
- **Friction-free security:** Payments do not require sign-up, login or multi-factor authentication
- **Privacy-preserving:** payment requires/shares no personal information
- **Bank account:** not required

TALER

Merchant Impact of Taler

- **Instant clearance:** one-click transactions and instant clearance at par
- **Easy & compliant:** GDPR & PCI-DSS compliance-free and without any effort
- **Major profit increase:** efficient protocol + no fraud = extremely low costs
- **1-click checkout:** without Amazon and without false positives in fraud detection

Taler: Unique Regulatory Features for Central Banks

https://www.snb.ch/en/mmr/papers/id/working_paper_2021_03

- Central bank issues digital coins equivalent to issuing cash
  ⇒ monetary policy remains under CB control
- Architecture with consumer accounts at commercial banks
  ⇒ no competition for commercial banking (S&L)
  ⇒ CB does not have to manage KYC, customer support
- Withdrawal limits and denomination expiration
  ⇒ protects against bank runs and hoarding
- Income transparency and possibility to set fees
  ⇒ additional insights into economy and new policy options
- Revocation protocols and loss limitations
  ⇒ exit strategy and handles catastrophic security incidents
- Privacy by cryptographic design not organizational compliance
  ⇒ CB cannot be forced to facilitate mass-surveillance

TALER

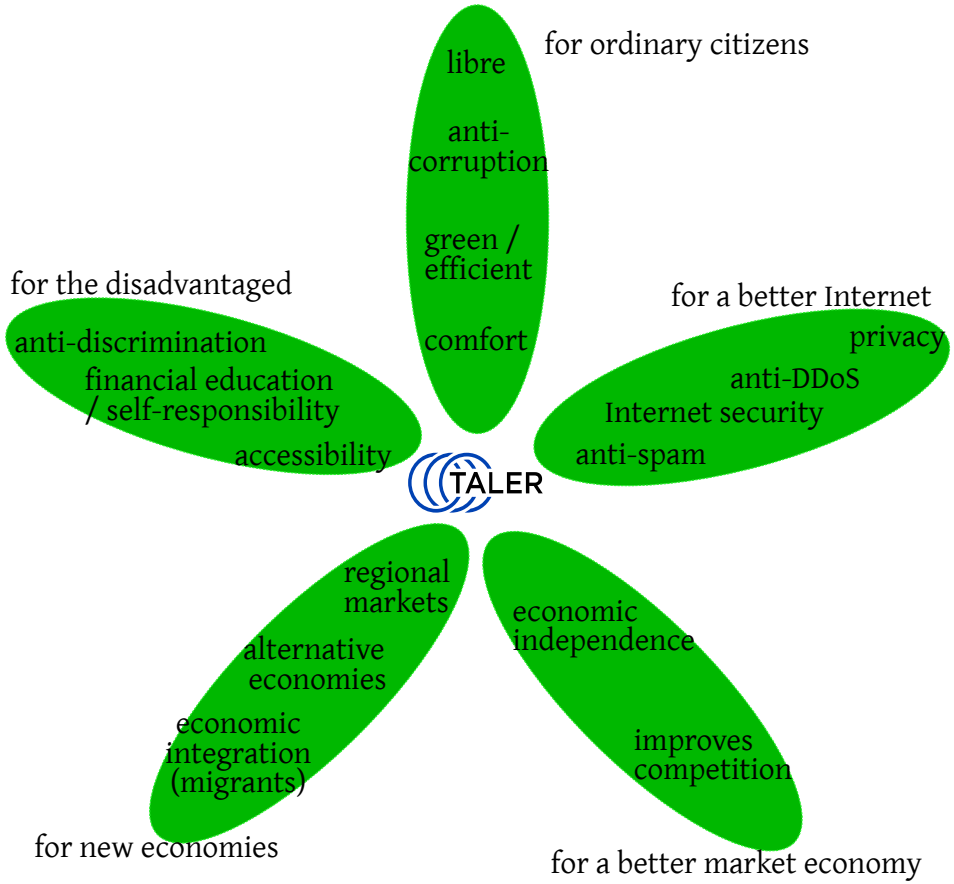
Usability of Taler

`https://demo.taler.net/`

1. Install browser extension.
2. Visit the `bank.demo.taler.net` to withdraw coins.
3. Visit the `shop.demo.taler.net` to spend coins.

TALER

Social Impact of Taler



TALER

Use Case: Journalism

Today:

- Corporate structure
- Advertising primary revenue
- Tracking readers critical for business success
- Journalism and marketing hard to distinguish

Use Case: Journalism

Today:

- Corporate structure
- Advertising primary revenue
- Tracking readers critical for business success
- Journalism and marketing hard to distinguish

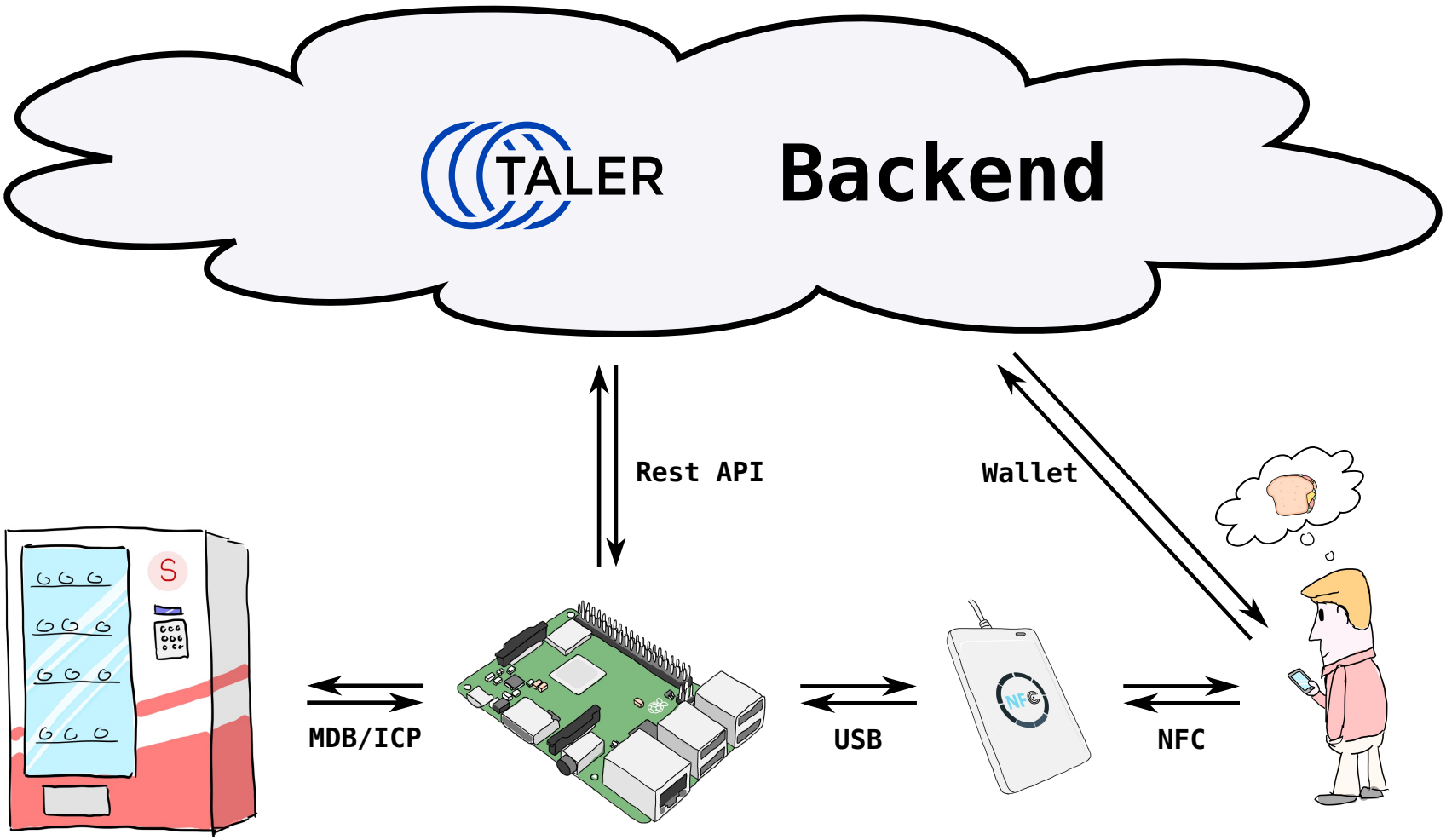
With GNU Taler:

- One-click micropayments per article
- Hosting requires no expertise
- Reader-funded reporting separated from marketing
- Readers can remain anonymous

TALER

Example: The Taler Snack Machine[2]

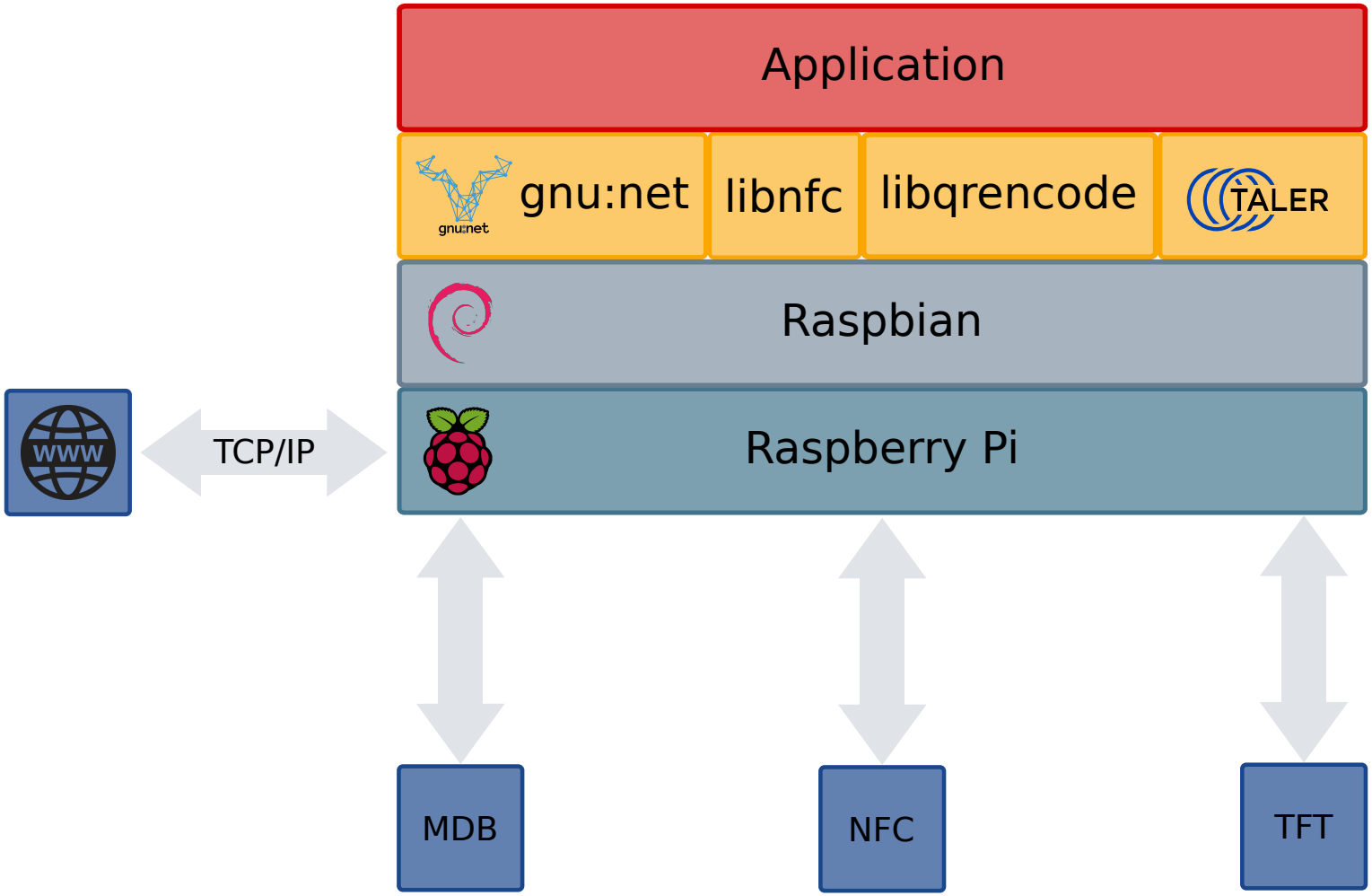
Integration of a MDB/ICP to Taler gateway.
Implementation of a NFC or QR-Code to Taler wallet interface.



[2]by M. Boss and D. Hofer

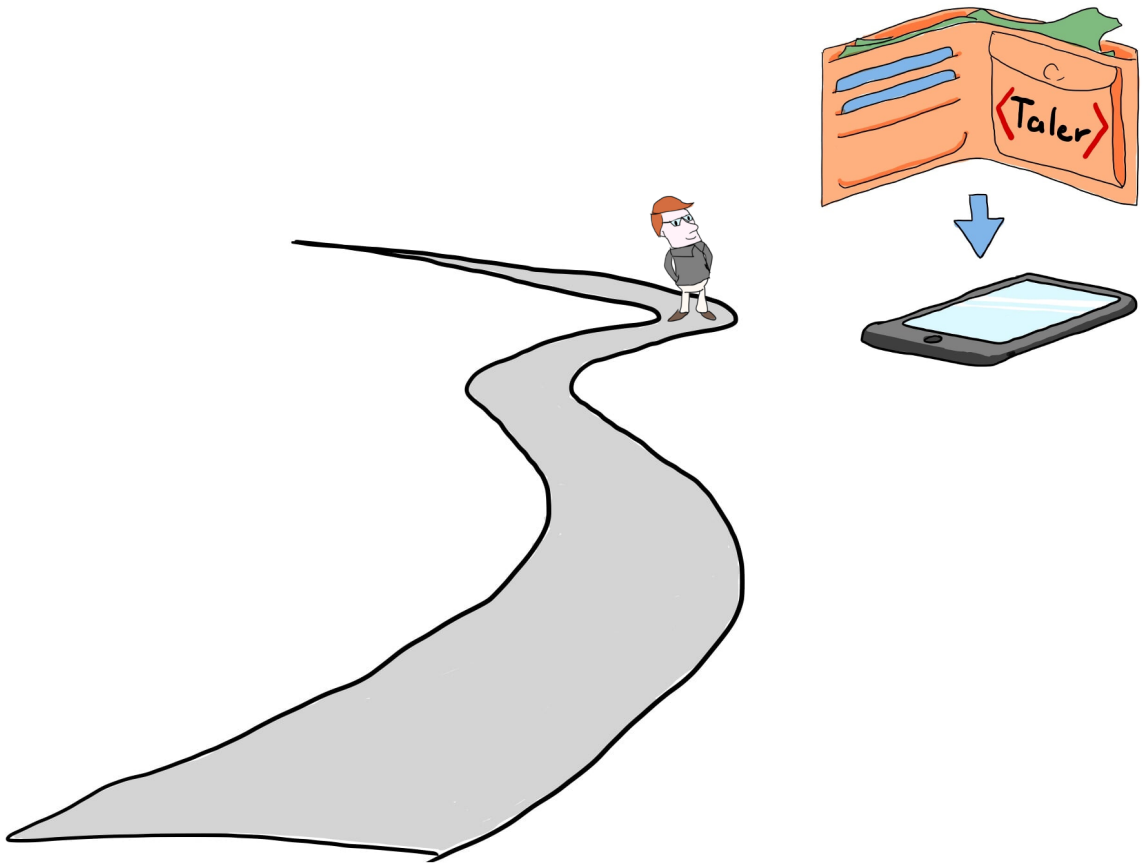
Software architecture for the Taler Snack Machine

Code at `https://git.taler.net/taler-mdb`

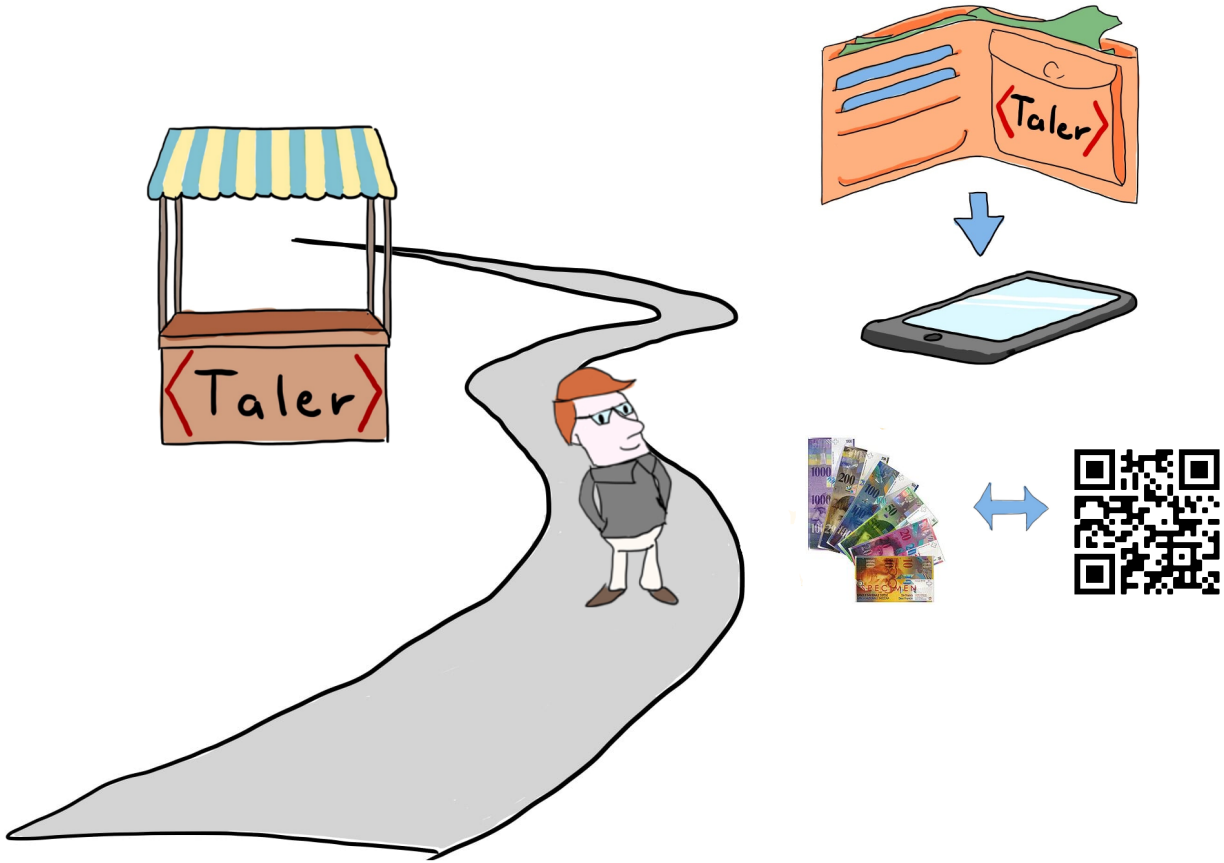


User story: Install App on Android

https://wallet.taler.net/



User story: Withdraw e-cash



User story: Use machine!

Real-world use

Component Zoo

The Taler Software Ecosystem: Overview

https://taler.net/en/docs.html

Taler is based on modular components that work together to provide a complete payment system:

- **Exchange:** Service provider for digital cash
  - Core exchange software (cryptography, database)
  - Air-gapped key management, real-time **auditing**
  - **libeufin**: Modular integration with banking systems
  - **challenger**: KYC service with OAuth 2.0 API
- **Merchant:** Integration service for existing businesses
  - Core merchant backend software (cryptography, database)
  - **Back-office interface** for staff
  - **Frontend integration** (E-commerce, Point-of-sale)
- **Wallet:** Consumer-controlled applications for e-cash
  - Multi-platform wallet software (for browsers & mobile phones)
  - Wallet backup storage providers (**sync** & **Anastasis**)

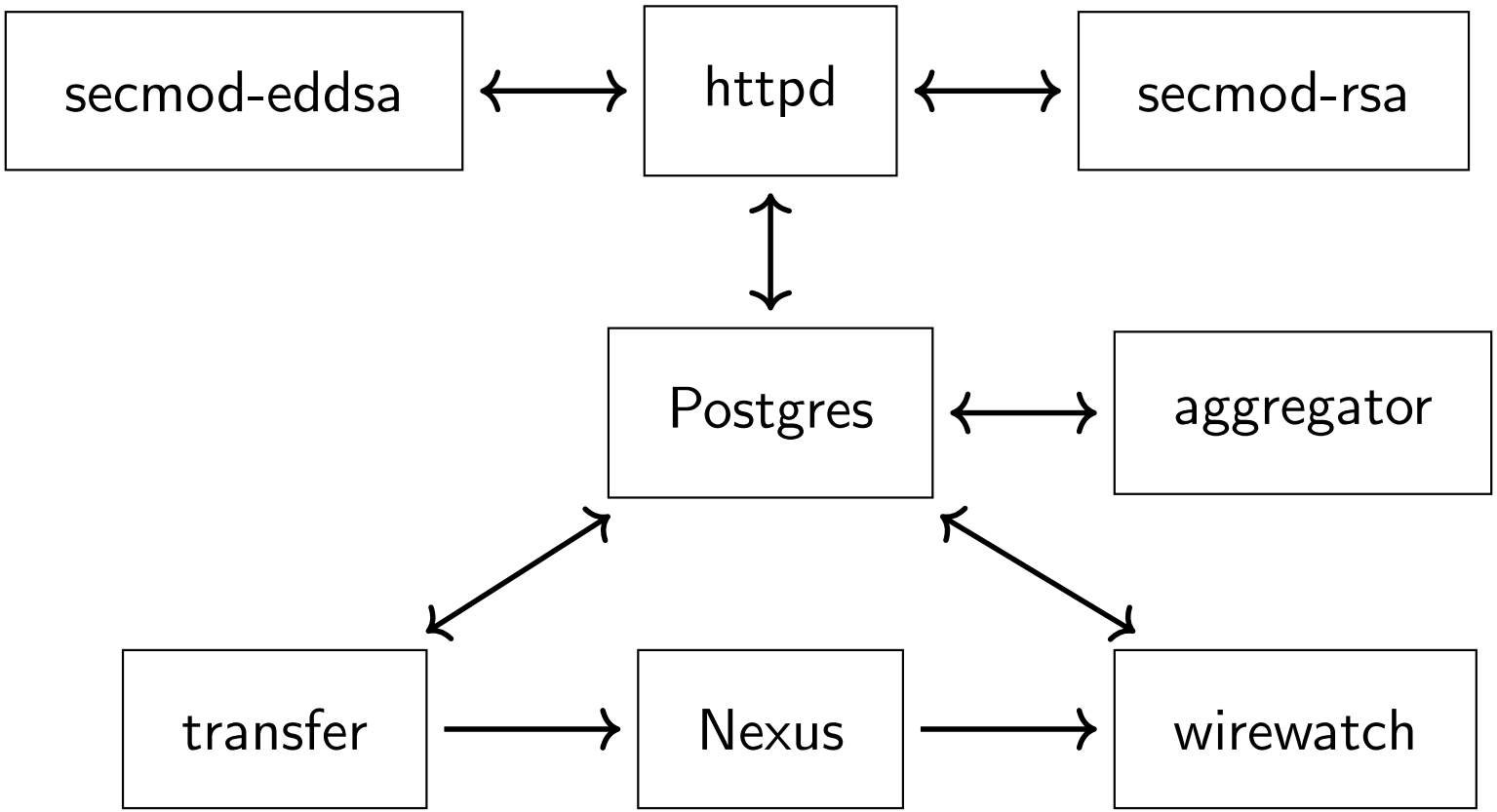
TALER

Taler Exchange

The **Exchange** is the core logic of the payment system.

- One exchange at minimum must be operated per currency
- Offers a REST API for merchants and customers
- Uses several helper processes for configuration and to interact with RTGS and cryptography
- KYC support via OAuth 2.0, KycAID or Persona APIs
- Implemented in C on top of GNU libmicrohttpd

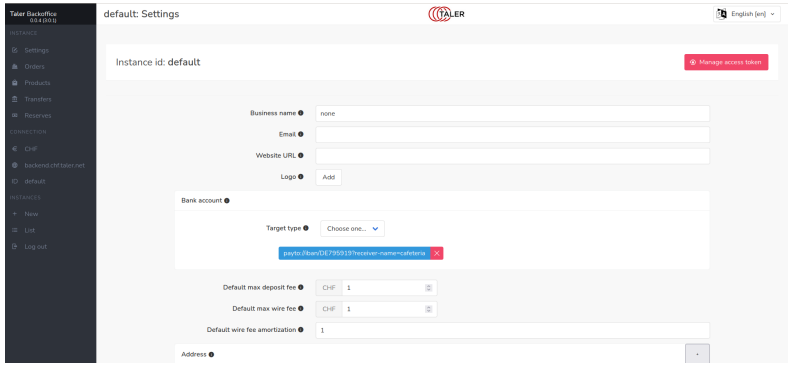
Taler: Exchange Architecture



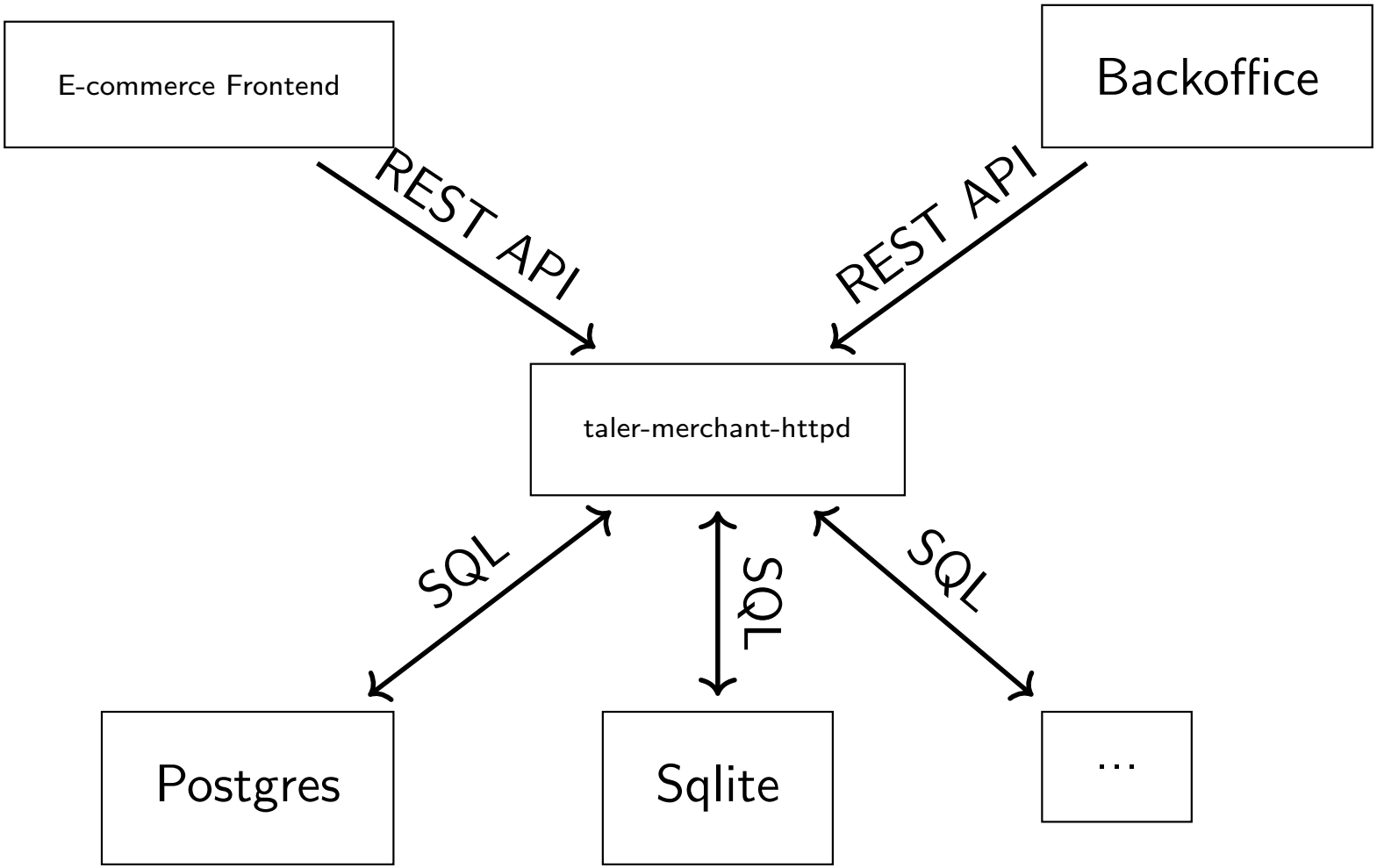
Taler Merchant

The **Merchant** is the software run by merchants to accept GNU Taler payments.

- REST API for integration with e-commerce
- SPA provides Web interface for administration
- Features include:
  - Multi-tenant support
  - Refunds
  - Templates
  - Webhooks
  - Inventory management (optional)
- Implemented in C on top of GNU libmicrohttpd



Taler: Merchant Perspective

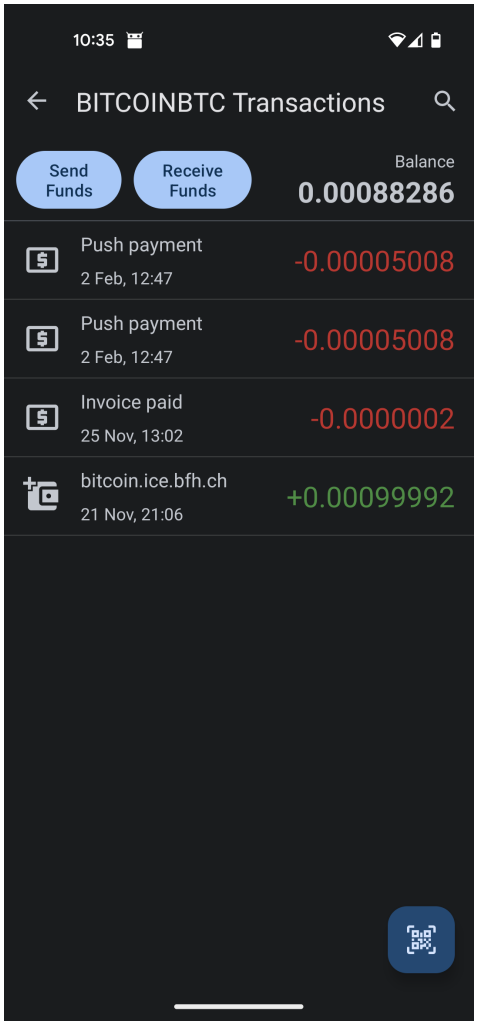


Taler Wallet

The **Wallet** is the software run by consumers to store their digital cash and authorize transactions.

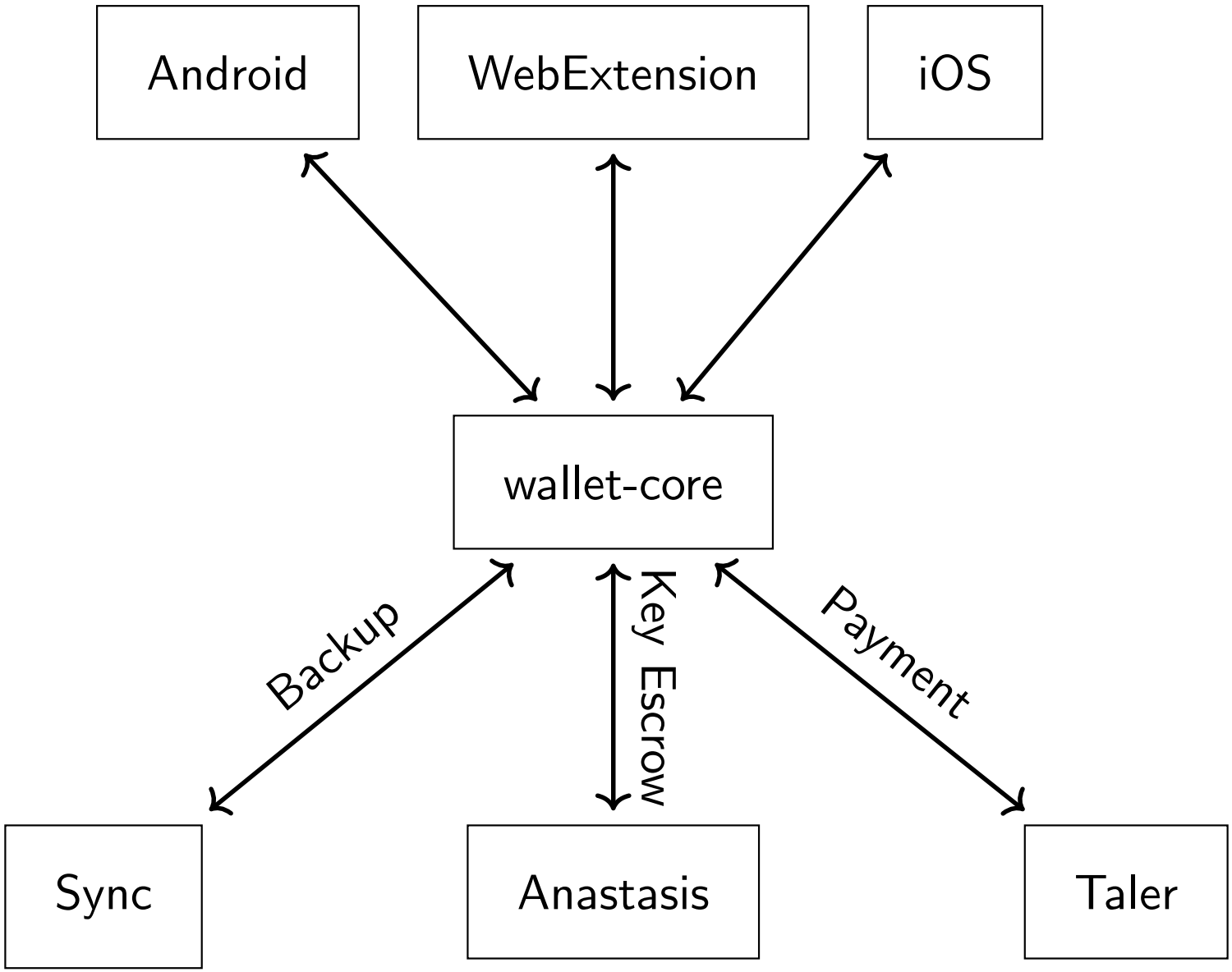
- **wallet-core** is the logic shared by all interfaces
- Works on Android, F-Droid, iOS, Ubuntu Touch, WebExtension (Chrome, Chromium, Firefox, etc.)
- Features include:
  - Multi-currency support
  - Wallet-to-wallet payments (NFC or QR code)
  - CRDT-like data model
- **wallet-core** implemented in TypeScript

Can be integrated into other Apps if desired.



Taler: Wallet Architecture

Background: `https://anastasis.lu/`

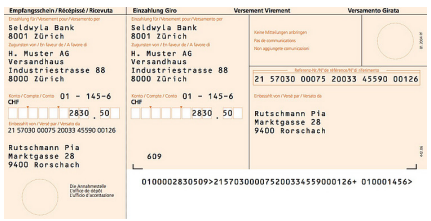


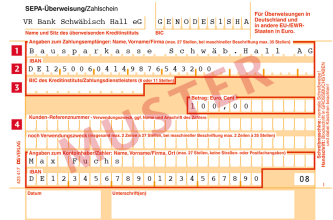
RFC 8905: `payto:` Uniform Identifiers for Payments and Accounts

Like `mailto:`, but for bank accounts instead of email accounts!

```
payto://<PAYMENT-METHOD>/<ACCOUNT-NR>
  ?subject=InvoiceNr42
  &amount=EUR:12.50
```

Default action: Open app to review and confirm payment.





TALER

Benefits of payto://

- Standardized way to represent financial resources (bank account, bitcoin wallet) and payments to them
- Useful on the client-side on the Web and for FinTech backend applications
- Payment methods (such as IBAN, ACH, Bitcoin) are registered with IANA and allow extra options

Taler wallet can generate payto://-URI for withdraw!

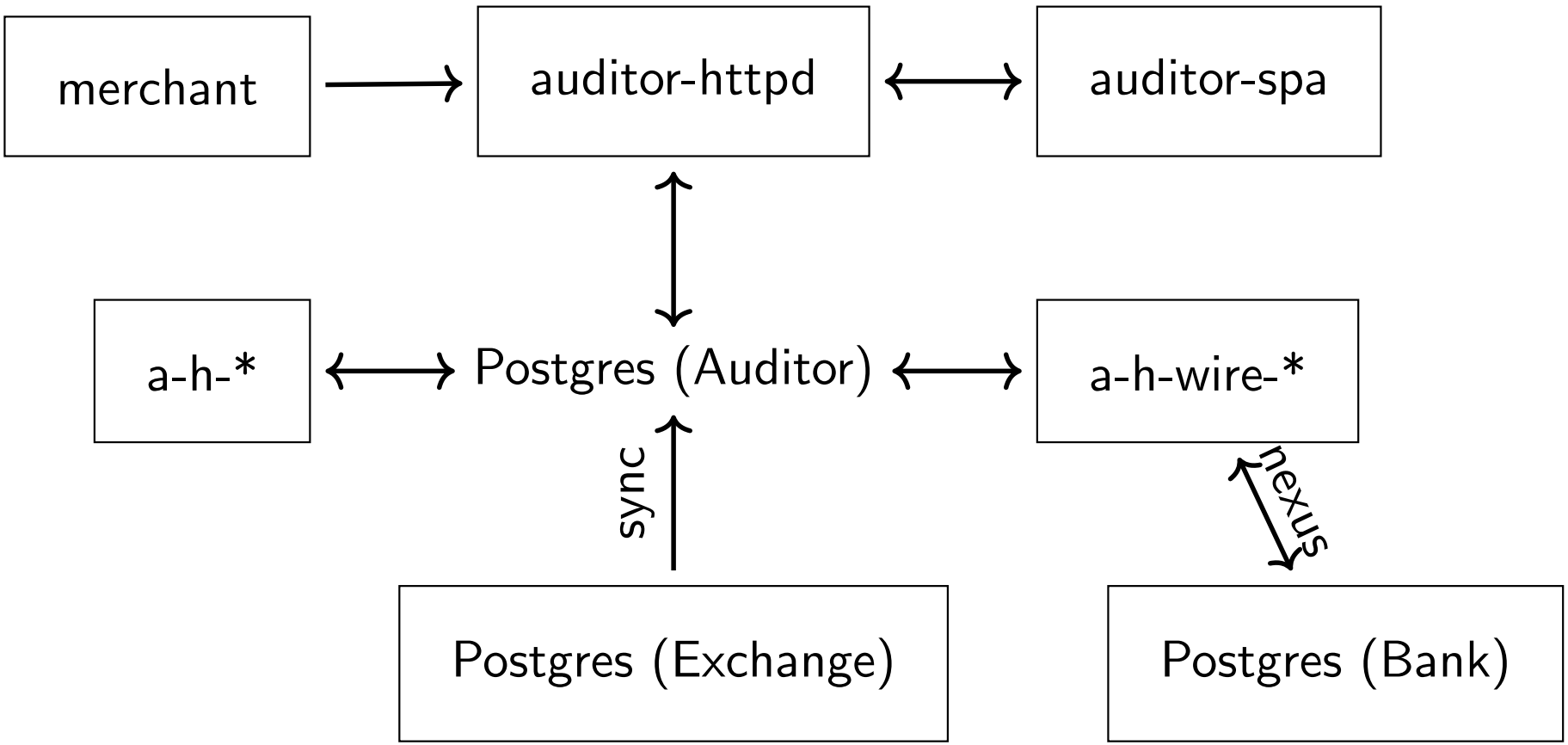
Taler Auditor

The **Auditor** is the software run by an independent auditor to validate the operation of an Exchange.

- REST API for additional report inputs by merchants (optional)
- Secure database replication logic
- Implemented in C on top of GNU libmicrohttpd

TALER

Taler: Auditor Perspective



libeufin-nexus

libeufin-nexus allows Taler components to interact with a core banking system. It:

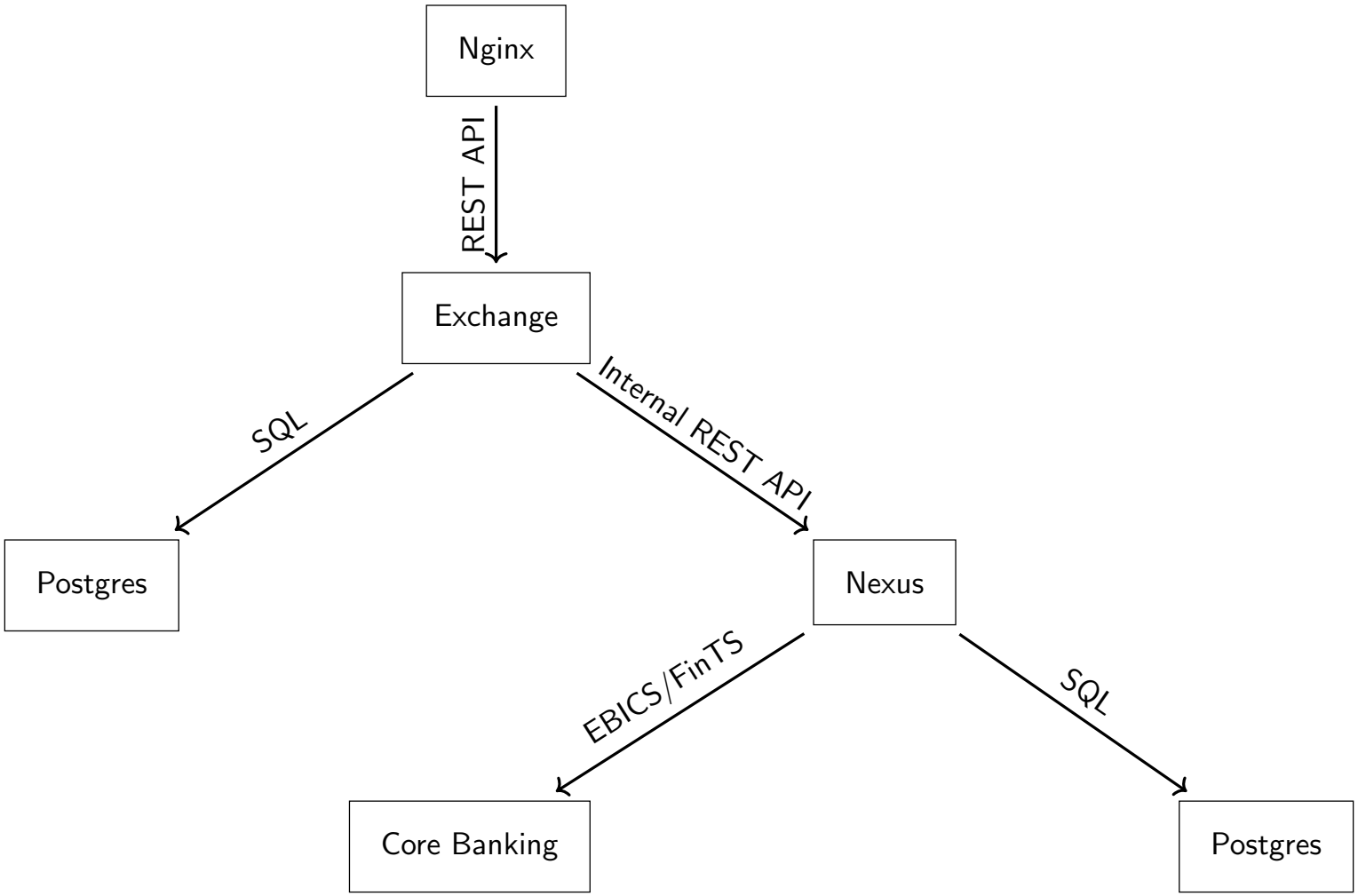
- provides an implementation of the Wire Gateway for the exchange
- supports EBICS 2.5 and 3.0
- other APIs such as FinTS or PSD2-style XS2A APIs can be added without requiring changes to the Exchange
- was tested with GLS Bank (DE) and Postfinance (CH) accounts and real EUR/CHF

libeufin-bank

libeufin-bank implements a standalone bank with a Web interface. It:

- provides the Taler Core Bank API for RESTful online banking using a Web interface (with multi-factor authentication)
- includes a Taler Wire Gateway for the exchange
- offers the Taler Bank Integration API to allow wallets to easily withdraw digital cash
- optionally provides the Taler Conversion Info API for currency conversion between fiat and regional currencies
- optionally integrates with libeufin-nexus to interact with a core banking system

Taler: Bank Perspective



Challenger

Challenger allows clients to obtain validated address (KYC) data about users:

- Customizable Web-based process for address validation
- Can validate phone numbers, e-mail addresses or physical mailing addresses
- Provides an exchange-compatible OAuth 2.0 API

Depolymerization

Depolymerization is a bridge between GNU Taler and blockchains, making Taler a layer 2 system for crypto-currencies (like Lightning).

- provides an implementation of the Wire Gateway for the exchange
- Works on top of Bitcoin and Ethereum crypto-currencies, with the DLTs as the "RTGS"
- Provides same API to Exchange as libeufin-nexus
- Implemented in Rust

`https://bitcoin.ice.bfh.ch/`

TALER

Pretix Taler payment plugin

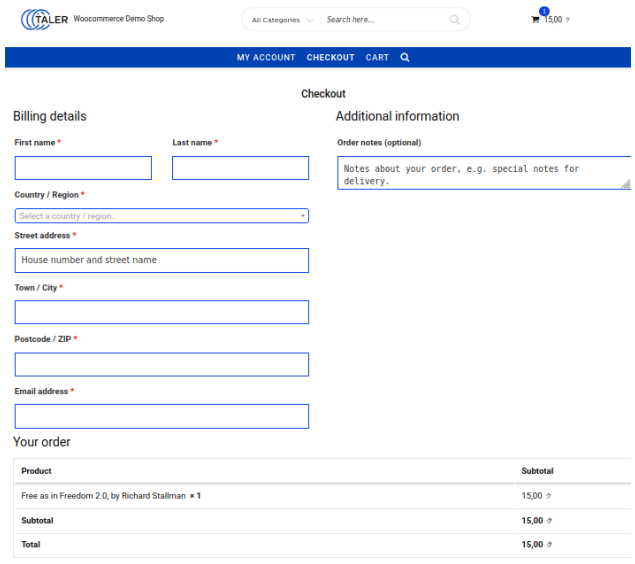
Pretix is a ticket sales system.

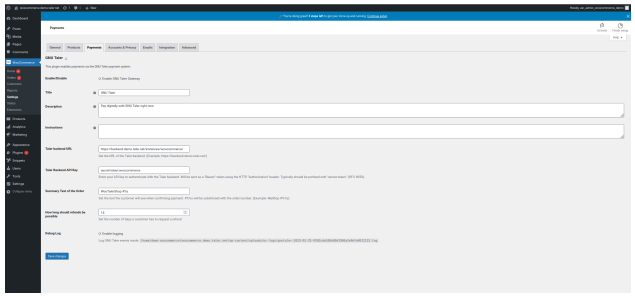
- Pretix payment plugin enables payments via GNU Taler
- Developed by Pretix.eu for €3,000 on behalf of Taler Systems SA

TALER

WooCommerce Taler payment plugin

- WooCommerce is an e-commerce plugin for WordPress.
- WooCommerce payment plugin enables payments via GNU Taler
- Features include:
  - Trivial configuration
  - Support for refunds
  - Full internationalization
- WooCommerce and its plugins are implemented in PHP





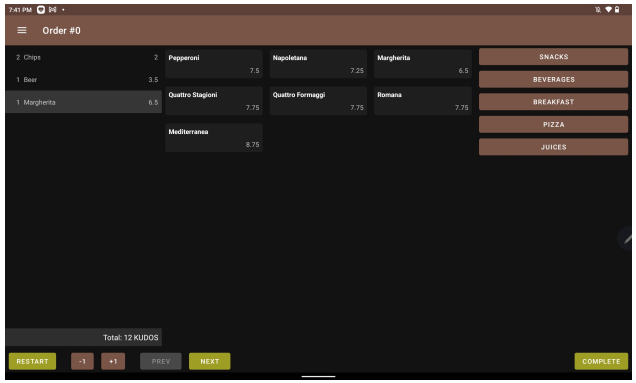
Joomla! Taler payment plugin

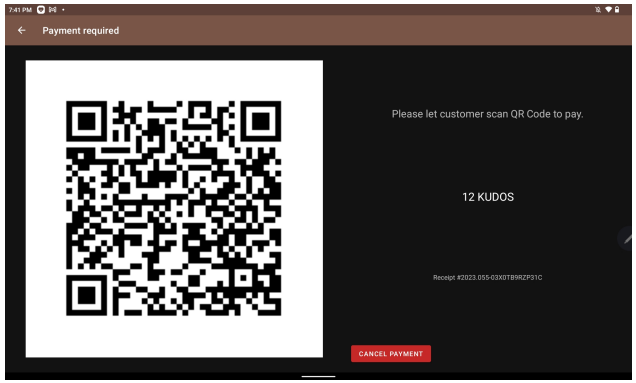
- Joomla! is an e-commerce platform
- Joomla! payment plugin enables payments via GNU Taler
- Features include:
  - Trivial configuration
  - Support for refunds
  - Full internationalization
- Joomla! and its plugins are implemented in PHP

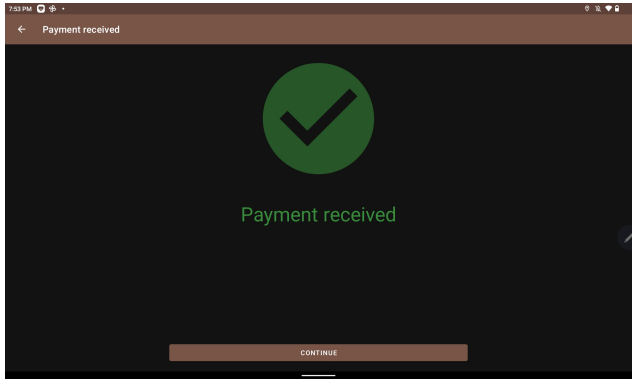
TALER

Point-of-Sale App for Android

- Allows merchant to generate orders against Taler backend and display QR code to enable customer to pay in person
- Patterned after ViewTouch restaurant UI
- Features include:
  - Internet-based configuration
  - Products sorted by categories
  - Easy undo of every operation
  - Manages multiple concurrent orders
- The Point-of-Sale App is implemented in Kotlin

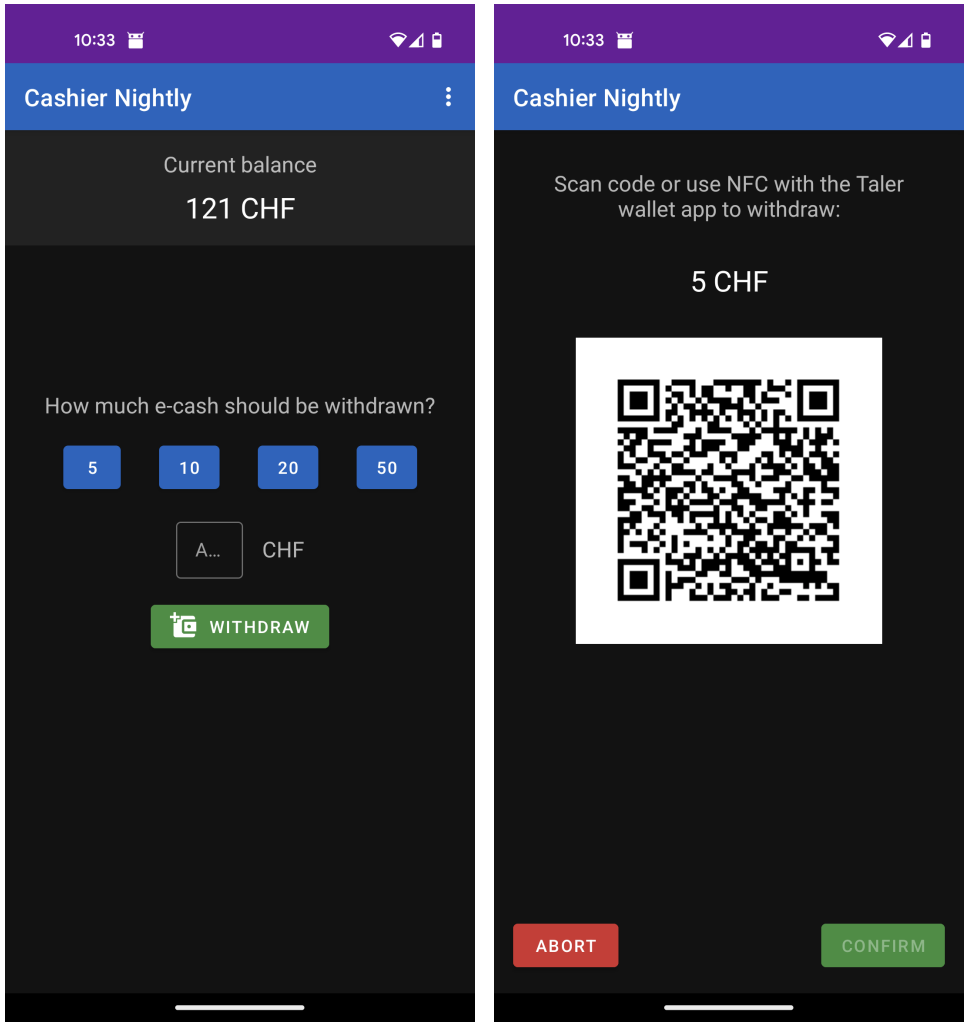




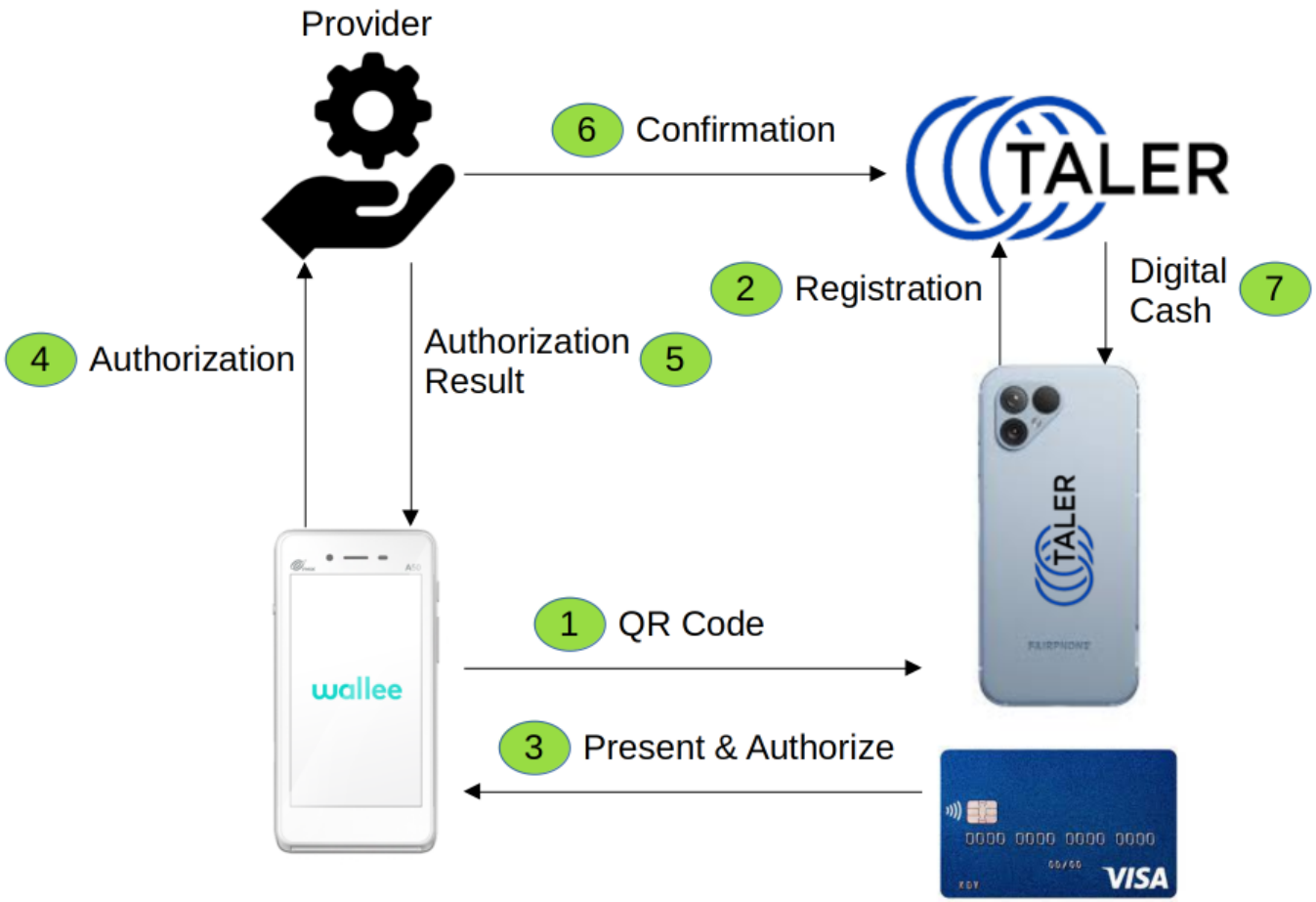


Cashier App for Android

- Enables BFH staff to convert cash to e-cash
- Staff has special bank accounts with limited funds
- Students can pay staff in cash to receive e-cash
- The Cashier App is implemented in Kotlin



Cashless2ecash by Joel Haeberli



TalDir (WiP)

TalDir is an extension to the existing peer-to-peer payment functionality.

- Registry to associate wallets with network addresses
- Extensible to different types of network services:
  - E-mail
  - SMS
  - Twitter
  - ...
- Send payments or invoices to wallets associated with network address
- Will **not** require sending wallet to use same network service

Protocol Basics

A Bachelor's Thesis Video

TALER

How does it work?

We use a few ancient constructions:

- Cryptographic hash function (1989)
- Blind signature (1983)
- Schnorr signature (1989)
- Diffie-Hellman key exchange (1976)
- Cut-and-choose zero-knowledge proof (1985)

But of course we use modern instantiations.

Definition: Taxability

We say Taler is taxable because:

- Merchant's income is visible from deposits.
- Hash of contract is part of deposit data.
- State can trace income and enforce taxation.

Definition: Taxability

We say Taler is taxable because:

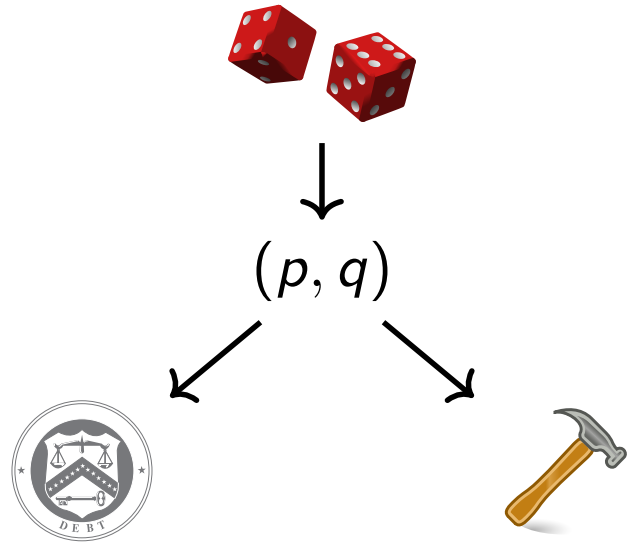
- Merchant's income is visible from deposits.
- Hash of contract is part of deposit data.
- State can trace income and enforce taxation.

Limitations:

- withdraw loophole
- *sharing* coins among family and friends

Exchange setup: Create a denomination key (RSA)

1. Generate random primes p, q.
2. Compute $n := pq$,
   $\phi(n) = (p-1)(q-1)$
3. Pick small $e < \phi(n)$ such that
   $d := e^{-1} \mod \phi(n)$ exists.
4. Publish public key (e, n).



Merchant: Create a signing key (EdDSA)

- Generate random number m mod o as private key
- Compute public key $M := mG$

$\downarrow$

m

$\downarrow$

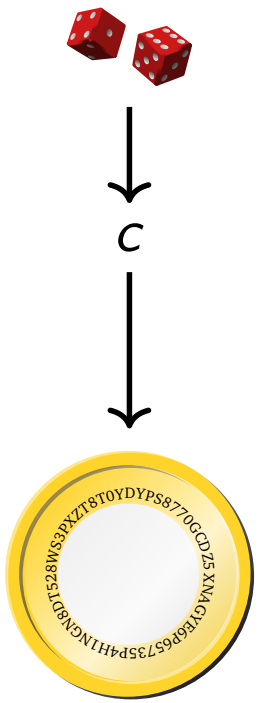
M

Capability: $m \Rightarrow$ 

TALER

Customer: Create a planchet (EdDSA)

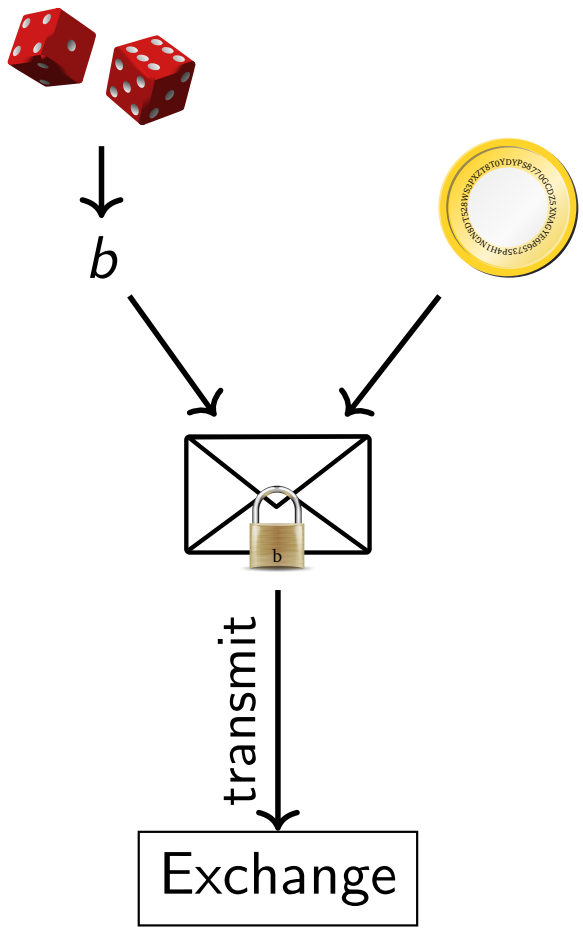
- Generate random number c mod o as private key
- Compute public key $C := cG$



Capability: $c \Rightarrow$ 

Customer: Blind planchet (RSA)

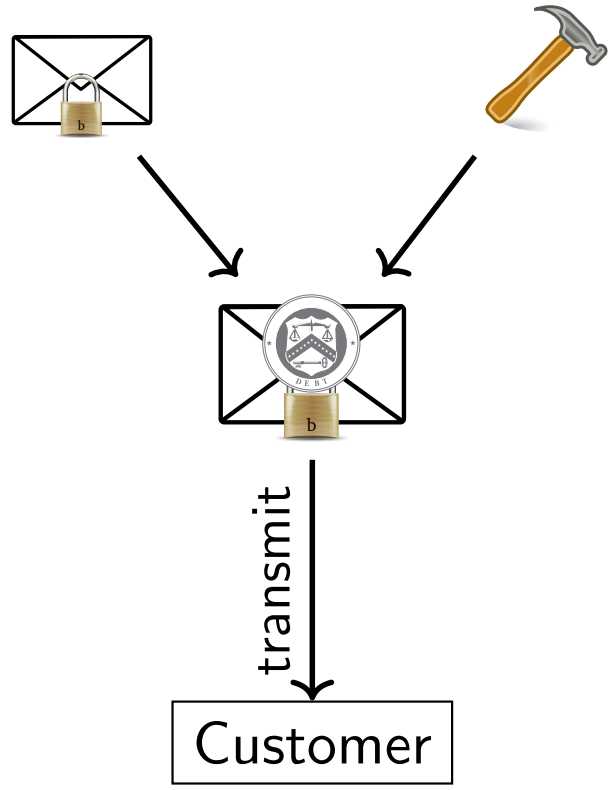
1. Obtain public key (e, n)
2. Compute $f := FDH(C)$, $f < n$.
3. Generate random blinding factor $b \in \mathbb{Z}_n$
4. Transmit $f' := fb^e \mod n$



TALER

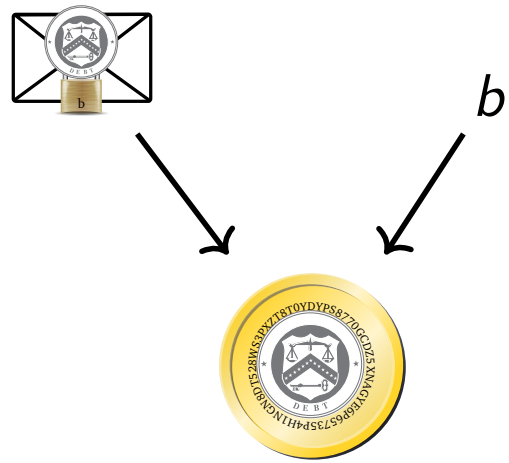
Exchange: Blind sign (RSA)

1. Receive f'.
2. Compute $s' := f'^d \mod n$.
3. Send signature s'.

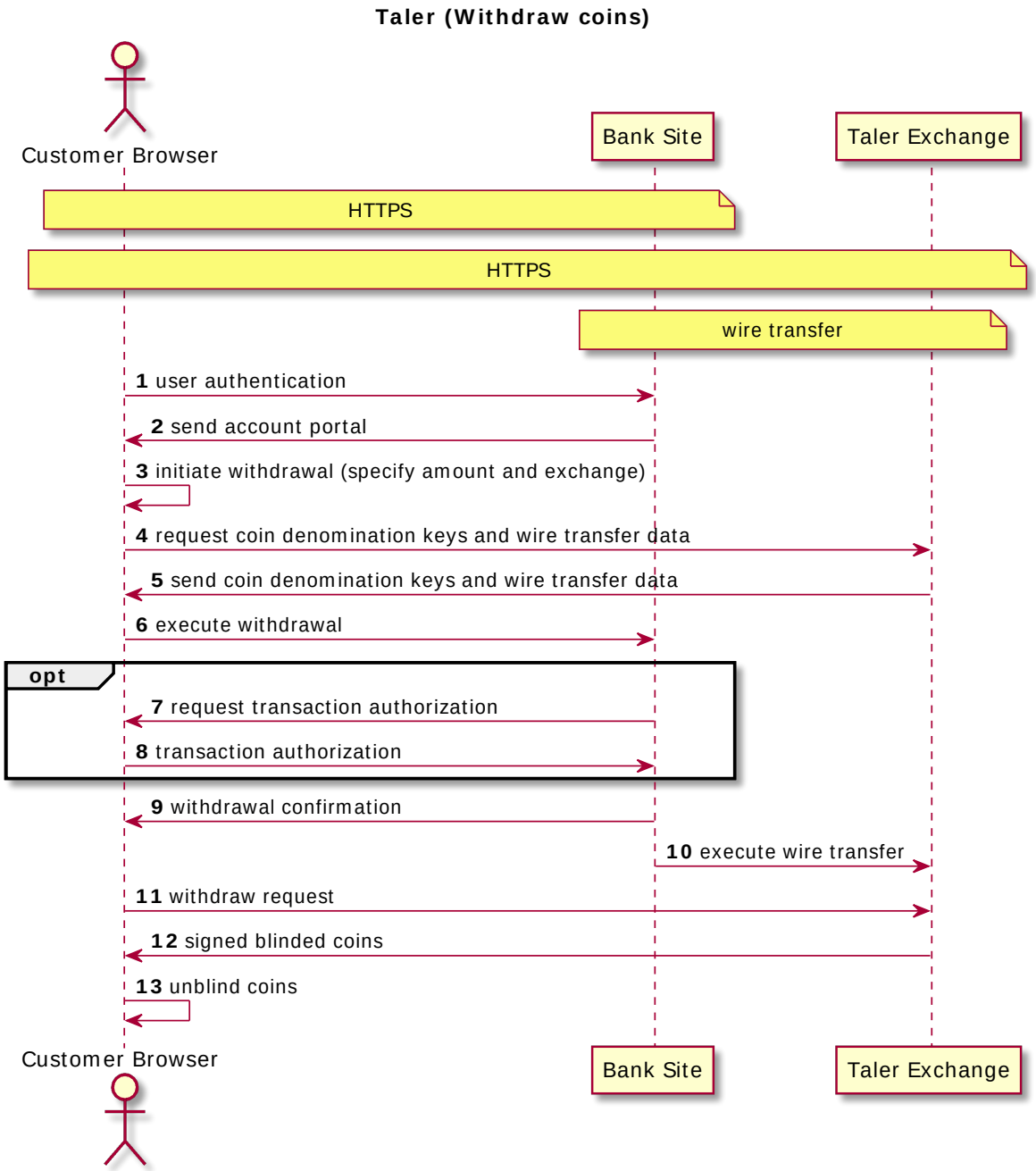


Customer: Unblind coin (RSA)

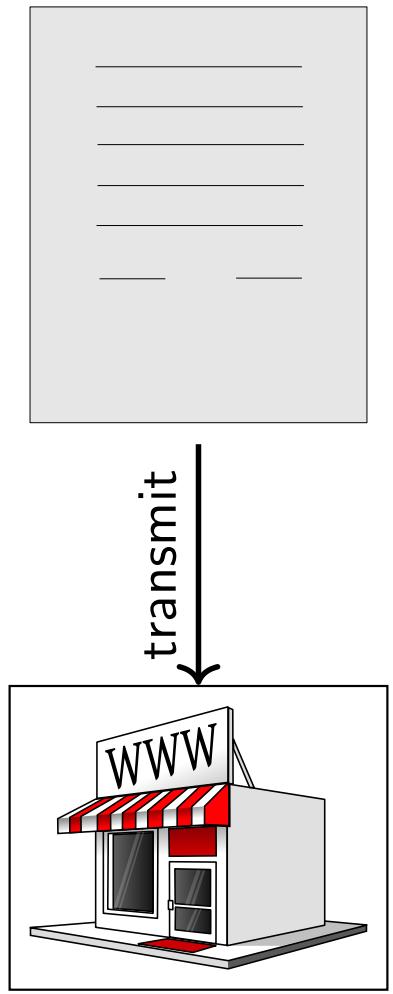
1. Receive s'.
2. Compute $s := s' b^{-1} \mod n$



Withdrawing coins on the Web



Customer: Build shopping cart



Merchant Integration: Contract


```
{
  "H_wire":"YTH0C4QBCQ10VDNTJN0DCTTV2Z6JHT5NF43F0RQHZ8JYB5NG4W4G...",
  "amount":{"currency":"EUR","fraction":0,"value":1},
  "max_fee":{"currency":"EUR","fraction":100000,"value":0},
  "auditors":[{"auditor_pub":"42V6TH91Q83FB846DK1GW3JQ5E8DS273W4..."}],
  "exchanges":[{"master_pub":"1T5FA8VQHMMKBHDMYPRZA2ZFK2S63AKF0Y...",
                "url":"https://exchange/"}],
  "fulfillment_url": "https://shop/article/42?tid=249&time=14714744",
  "merchant":{"address":"Mailbox␣4242","jurisdiction":"Jersey",
                "name":"Shop␣Inc."},
  "merchant_pub":"Y1ZAR5346J3ZTEXJCHQY9NJN78EZ2HSKZK8M0MYTNRJG5N...",
  "products":[{
    "description":"Essay:␣The␣GNU␣Project",
    "price":{"currency":"EUR","fraction":0,"value":1},
    "product_id":42,"quantity":1}],
  "pay_deadline":"/Date(1480119270)/",
  "refund_deadline":"/Date(1471522470)/",
  "timestamp":"/Date(1471479270)/",
  "transaction_id":249960194066269
}
```


TALER

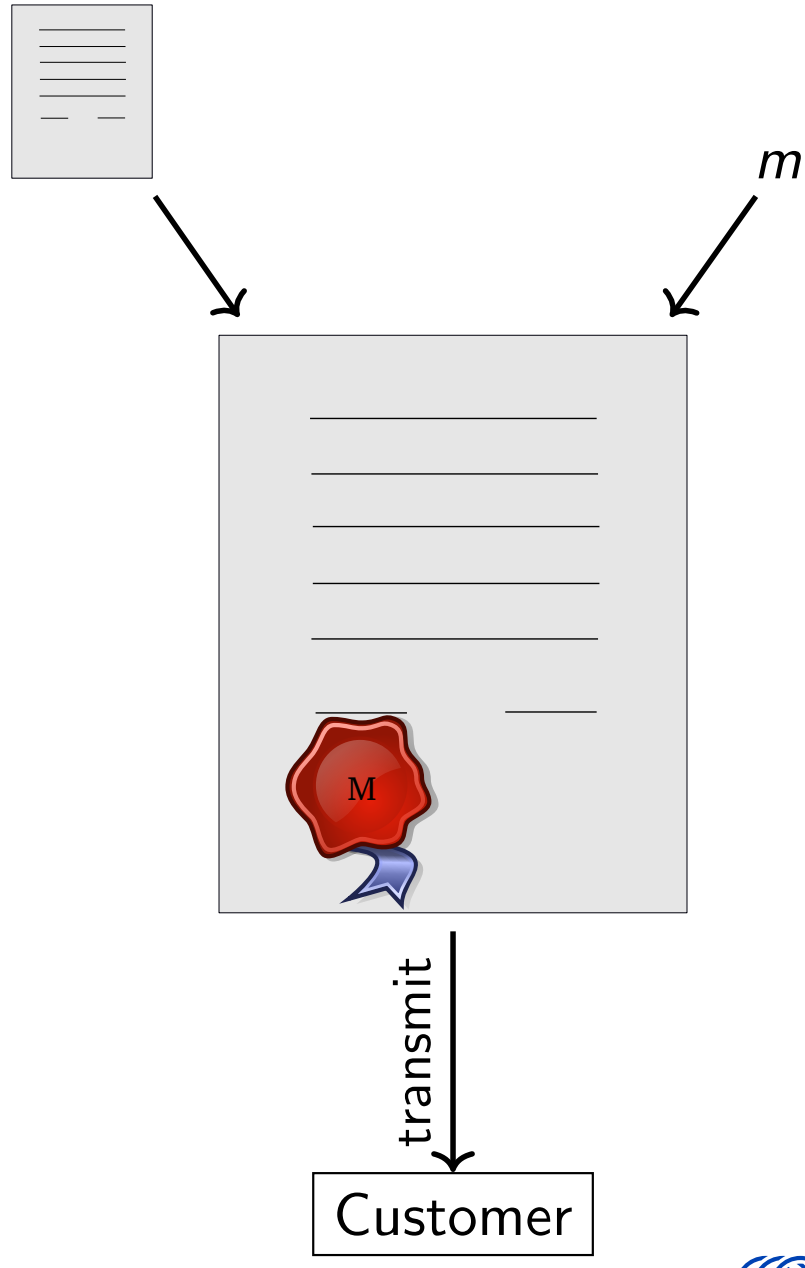
Merchant Integration: Payment Request

```
HTTP/1.1 402 Payment Required
Content-Type: text/html; charset=UTF-8
Taler: taler://pay/merchant.example.com/JYABKFIEWB

<!DOCTYPE html>
<html>
  <!-- Fallback page with QR code for browsers without
       Taler support / browser extension. -->
</html>
```

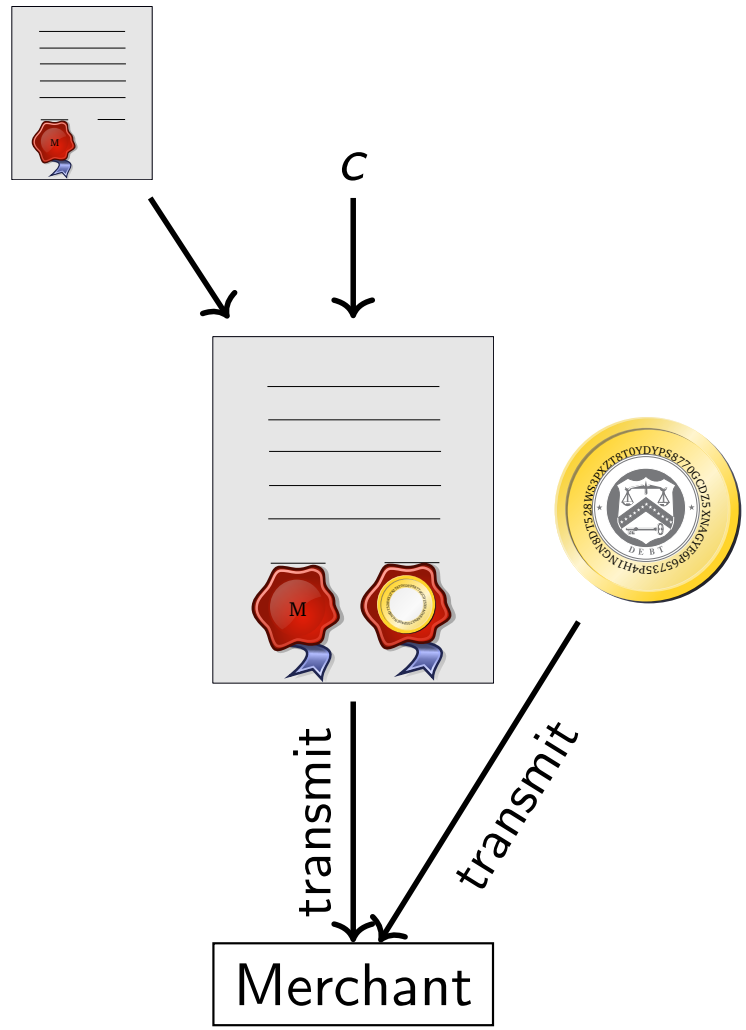
Merchant: Propose contract (EdDSA)

1. Complete proposal D.
2. Send D, $EdDSA_m(D)$



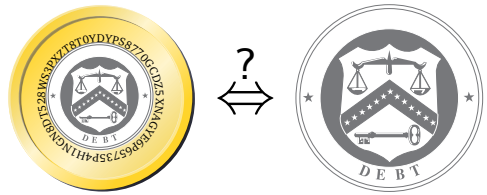
Customer: Spend coin (EdDSA)

1. Receive proposal D, $EdDSA_m(D)$.
2. Send s, C, $EdDSA_c(D)$



Merchant and Exchange: Verify coin (RSA)

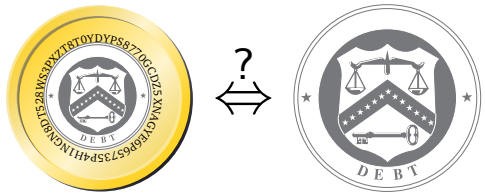
$$s^e \stackrel{?}{\equiv} FDH(C) \mod n$$



The exchange does not only verify the signature, but also checks that the coin was not double-spent.

Merchant and Exchange: Verify coin (RSA)

$$s^e \stackrel{?}{\equiv} FDH(C) \mod n$$

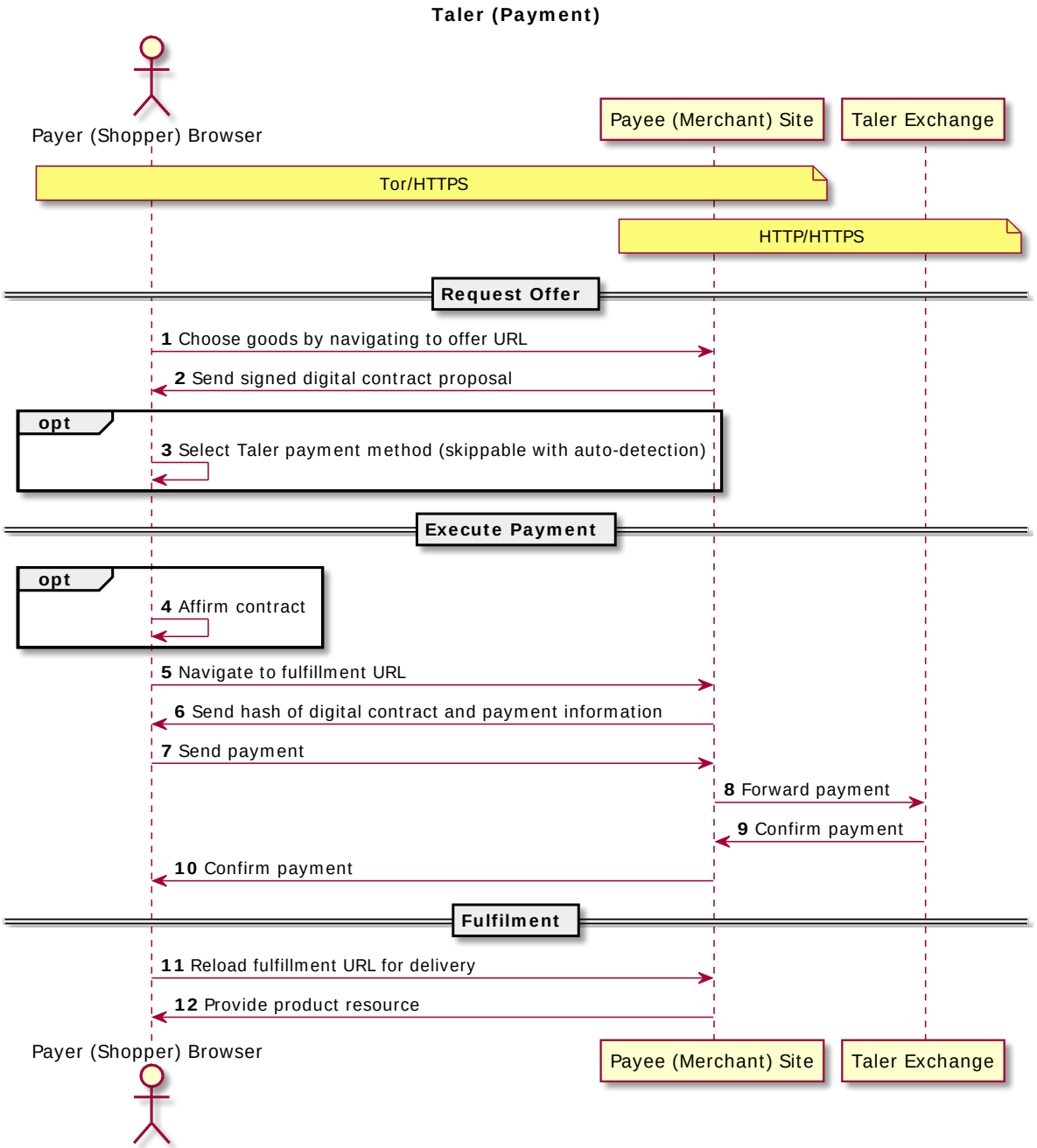


The exchange does not only verify the signature, but also checks that the coin was not double-spent.

Taler is an online payment system.

TALER

Payment processing with Taler



Giving change

It would be inefficient to pay EUR 100 with 1 cent coins!

- Denomination key represents value of a coin.
- Exchange may offer various denominations for coins.
- Wallet may not have exact change!
- Usability requires ability to pay given sufficient total funds.

Giving change

It would be inefficient to pay EUR 100 with 1 cent coins!

- Denomination key represents value of a coin.
- Exchange may offer various denominations for coins.
- Wallet may not have exact change!
- Usability requires ability to pay given sufficient total funds.

Key goals:

- maintain unlinkability
- maintain taxability of transactions

Giving change

It would be inefficient to pay EUR 100 with 1 cent coins!

- Denomination key represents value of a coin.
- Exchange may offer various denominations for coins.
- Wallet may not have exact change!
- Usability requires ability to pay given sufficient total funds.

Key goals:

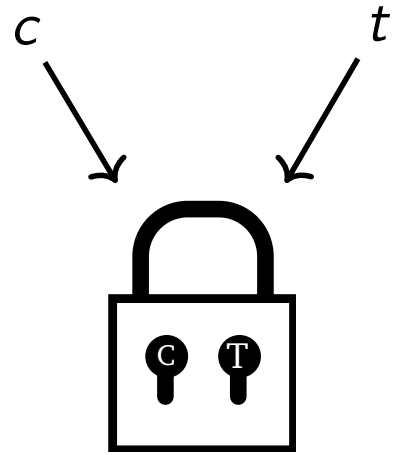
- maintain unlinkability
- maintain taxability of transactions

Method:

- Contract can specify to only pay *partial value* of a coin.
- Exchange allows wallet to obtain *unlinkable change* for remaining coin value.

Diffie-Hellman (ECDH)

1. Create private keys $c, t \mod o$
2. Compute $C := cG$
3. Compute $T := tG$
4. Compute DH
   $cT = c(tG) = t(cG) = tC$



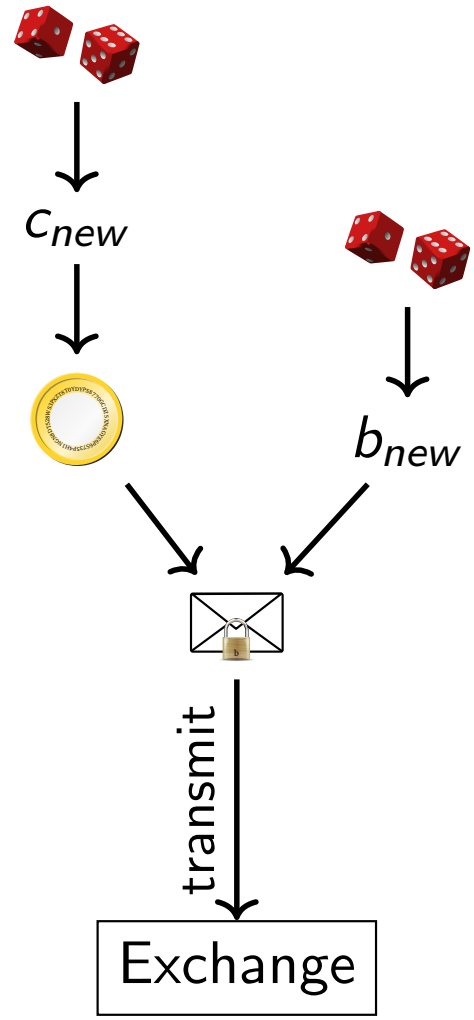
TALER

Strawman solution

Given partially spent private coin key c_{old}:

1. Generate random $c_{new} \mod o$ as private key
2. Compute public key $C_{new} = c_{new} G$
3. Generate random b_{new}
4. Compute $f_{new} := FDH(C_{new})$, $m < n$.
5. Transmit $f'_{new} := f_{new} b^e_{new} \mod n$

... and sign request for change with c_{old}.

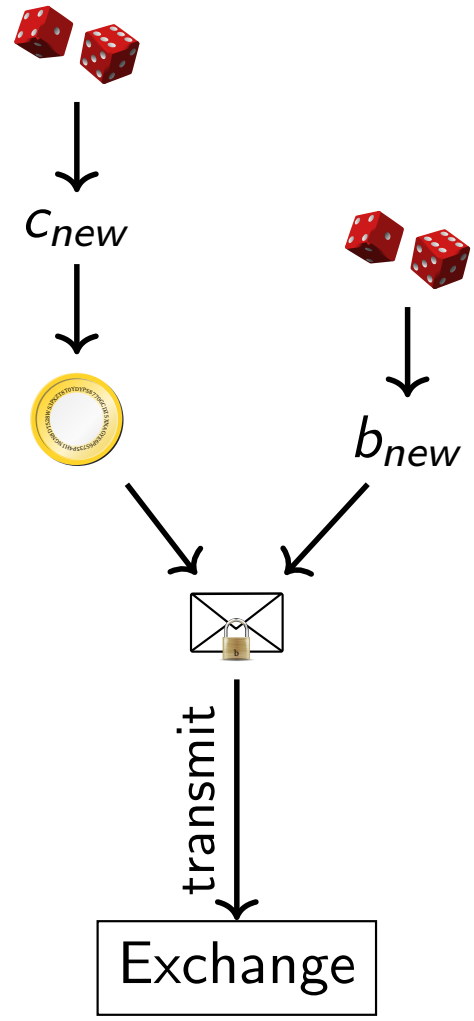


Strawman solution

Given partially spent private coin key c_{old}:

1. Generate random $c_{new} \mod o$ as private key
2. Compute public key $C_{new} = c_{new} G$
3. Generate random b_{new}
4. Compute $f_{new} := FDH(C_{new})$, $m < n$.
5. Transmit $f'_{new} := f_{new} b^e_{new} \mod n$

... and sign request for change with c_{old}.

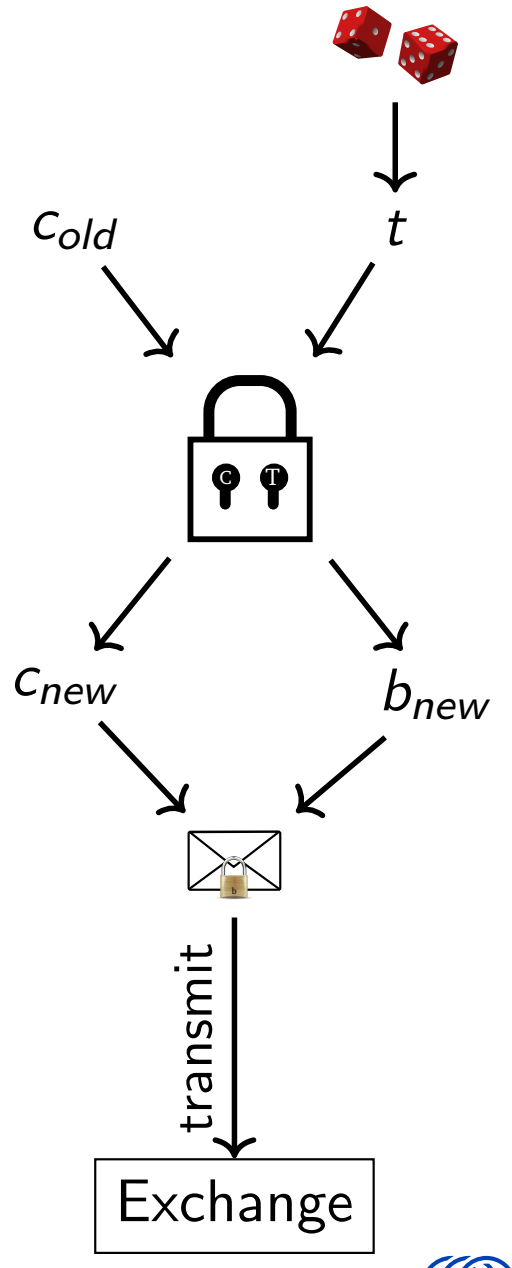


Problem: Owner of c_{new} may differ from owner of c_{old}!

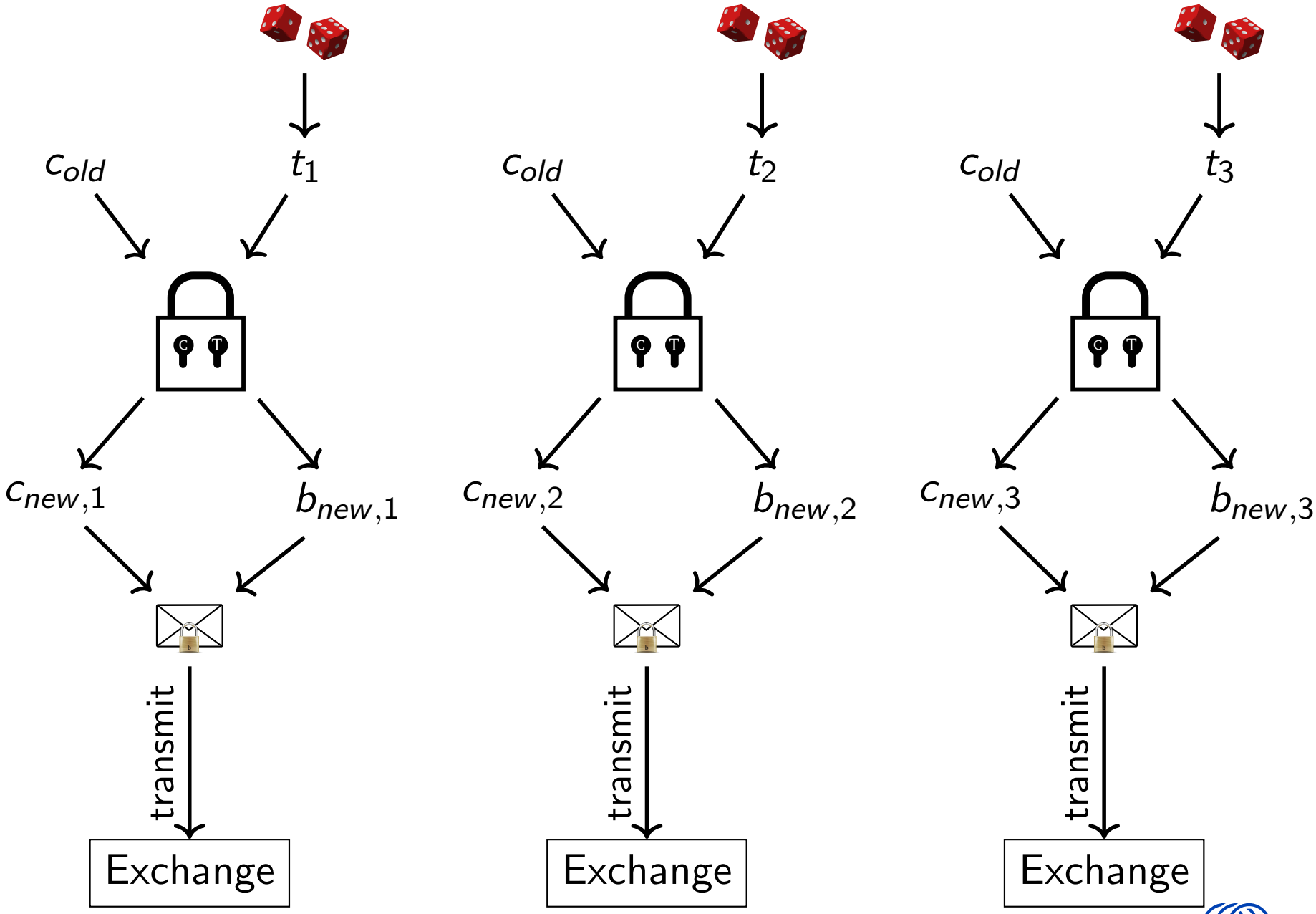
Customer: Transfer key setup (ECDH)

Given partially spent private coin key c_{old}:

1. Let $C_{old} := c_{old} G$ (as before)
2. Generate random private transfer key t mod o
3. Compute $T := tG$
4. Compute $X := c_{old}(tG) = t(c_{old} G) = tC_{old}$
5. Derive c_{new} and b_{new} from X
6. Compute $C_{new} := c_{new} G$
7. Compute $f_{new} := FDH(C_{new})$
8. Transmit $f'_{new} := f_{new} b^e_{new}$



Cut-and-Choose



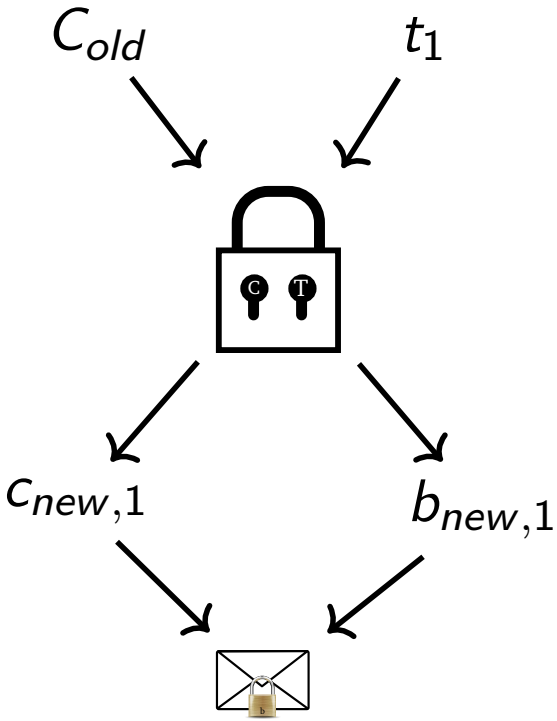
Exchange: Choose!

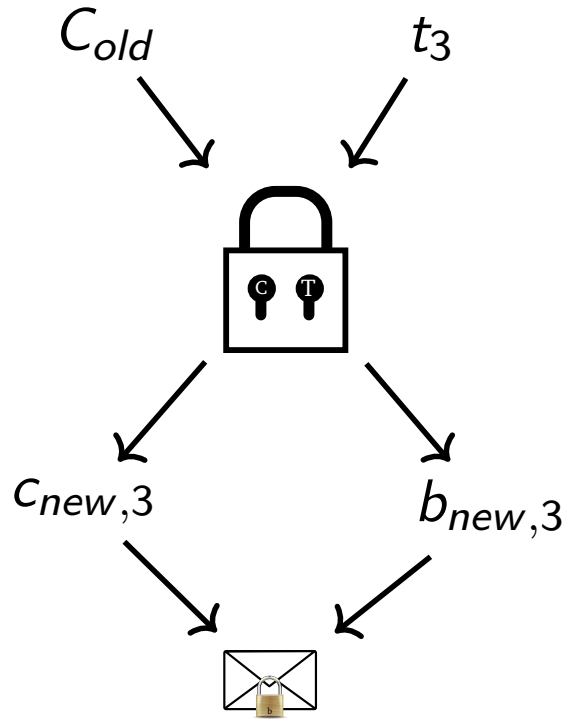
Exchange sends back random $\gamma \in \{1, 2, 3\}$ to the customer.

Customer: Reveal

1. If $\gamma = 1$, send t_2, t_3 to exchange
2. If $\gamma = 2$, send t_1, t_3 to exchange
3. If $\gamma = 3$, send t_1, t_2 to exchange

Exchange: Verify ($\gamma = 2$)

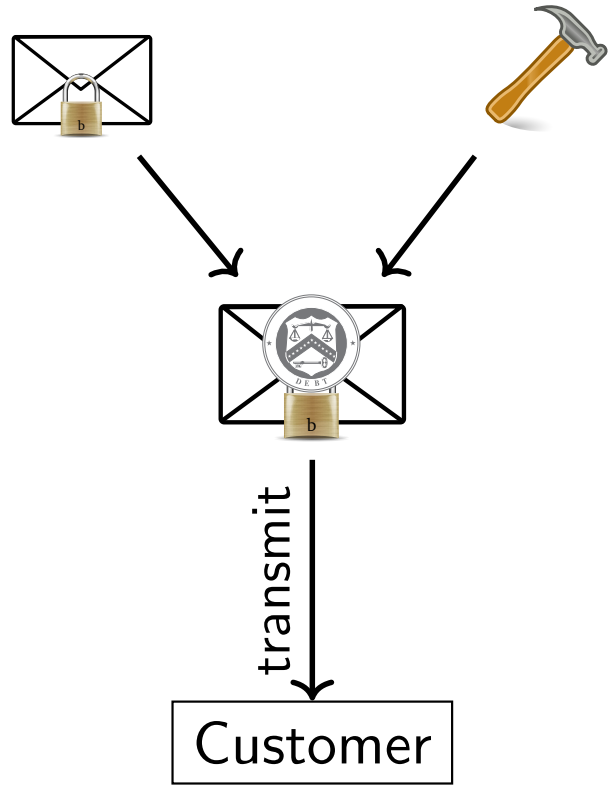




TALER

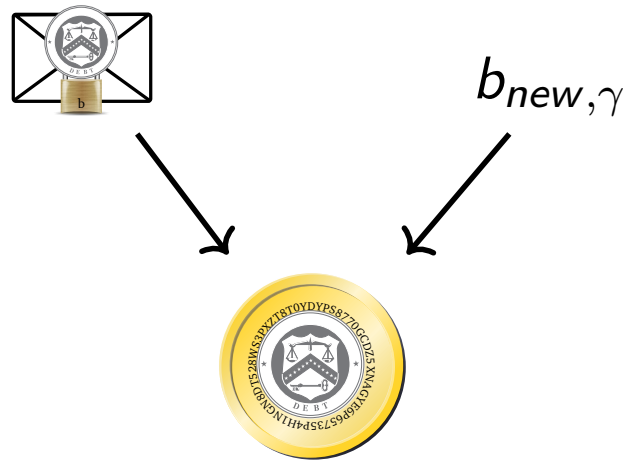
Exchange: Blind sign change (RSA)

1. Take $f'_{new,\gamma}$.
2. Compute $s' := f'^{d}_{new,\gamma} \mod n$.
3. Send signature s'.



Customer: Unblind change (RSA)

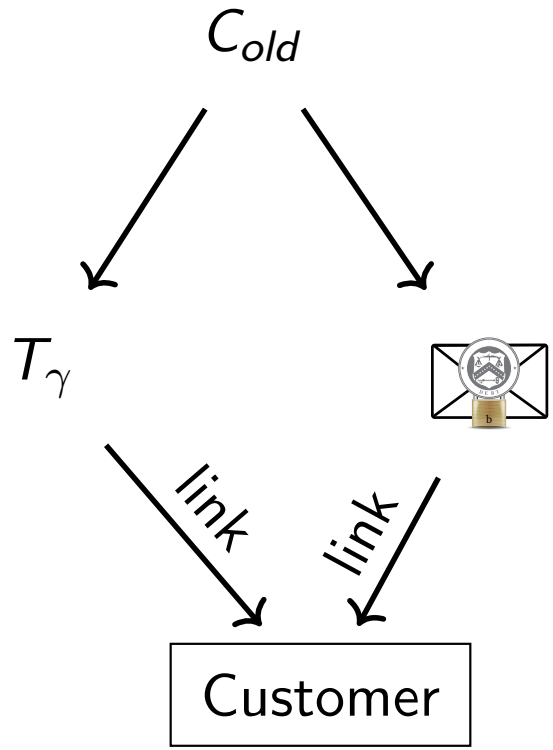
1. Receive s'.
2. Compute $s := s' b_{new,\gamma}^{-1} \mod n$.



Exchange: Allow linking change

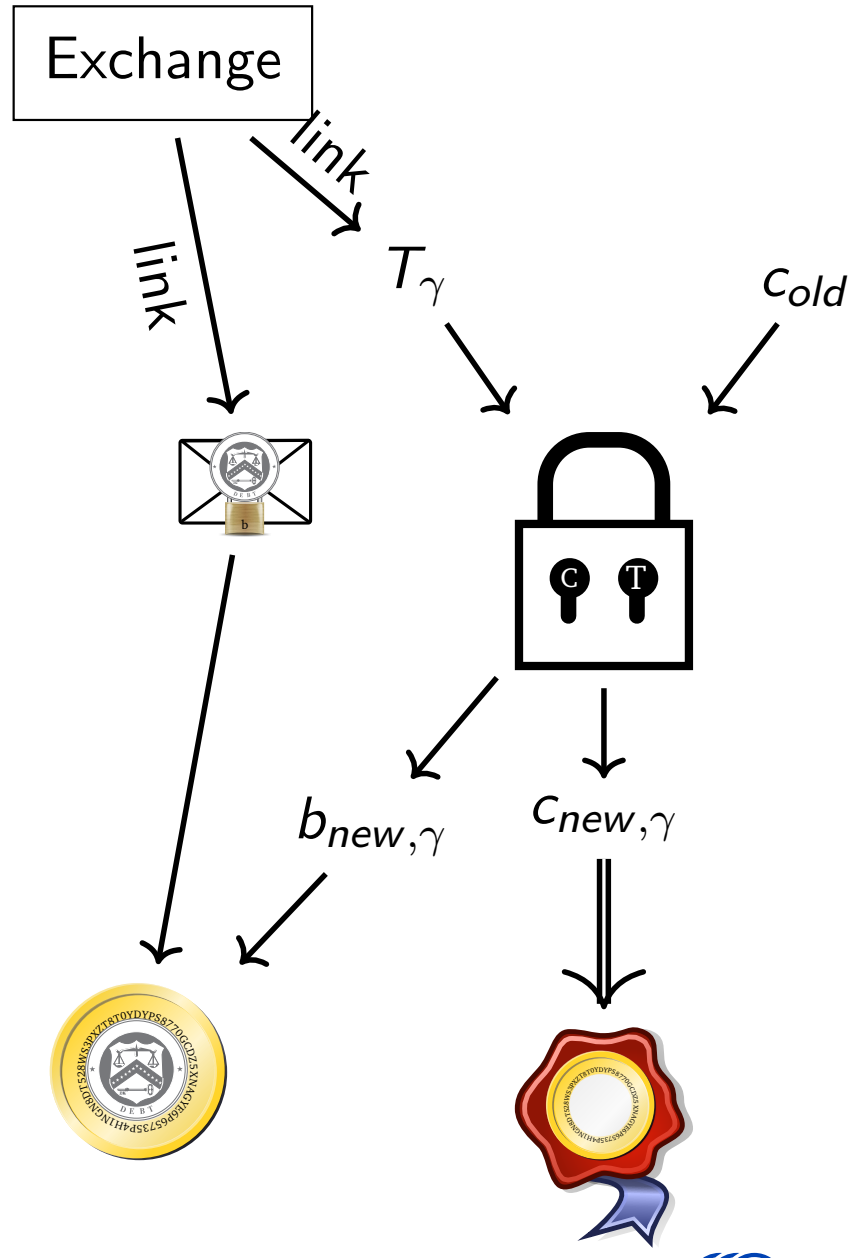
Given C_{old}

return T_γ, $s := s' b_{new,\gamma}^{-1} \mod n$.



Customer: Link (threat!)

1. Have c_{old}.
2. Obtain T_γ, s from exchange
3. Compute $X_\gamma = c_{old}\, T_\gamma$
4. Derive $c_{new,\gamma}$ and $b_{new,\gamma}$ from X_γ
5. Unblind $s := s' b_{new,\gamma}^{-1} \mod n$



Refresh protocol summary

- Customer asks exchange to convert old coin to new coin
- Protocol ensures new coins can be recovered from old coin

$\Rightarrow$ New coins are owned by the same entity!

Thus, the refresh protocol allows:

- To give unlinkable change.
- To give refunds to an anonymous customer.
- To expire old keys and migrate coins to new ones.
- To handle protocol aborts.

Transactions via refresh are equivalent to sharing a wallet.

Attacks & Defenses

Key management

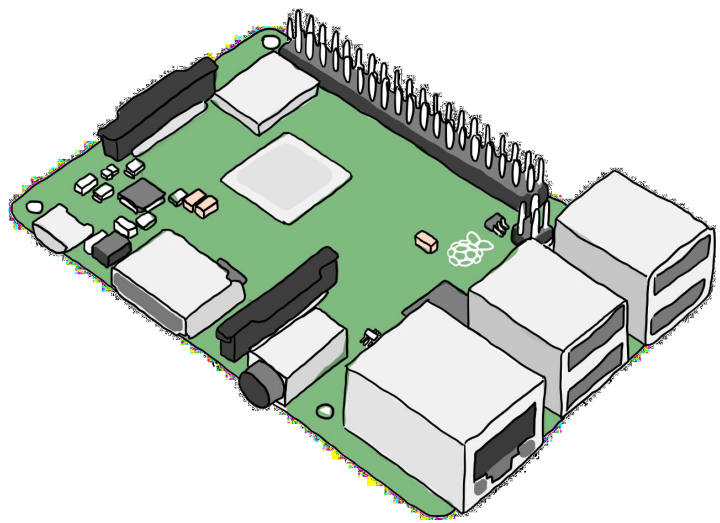
Taler has many types of keys:

- Coin keys
- Denomination keys
- Online message signing keys
- Offline key signing keys
- Merchant keys
- Auditor key
- Security module keys
- Transfer keys
- Wallet keys
- *TLS keys, DNSSEC keys*

Offline keys

Both exchange and auditor use offline keys.

- Those keys must be backed up and remain highly confidential!
- We recommend that computers that have ever had access to those keys to NEVER again go online.
- We recommend using a Raspberry Pi for offline key operations. Store it in a safe under multiple locks and keys.
- Apply full-disk encryption on offline-key signing systems.
- Have 3–5 full-disk backups of offline-key signing systems.

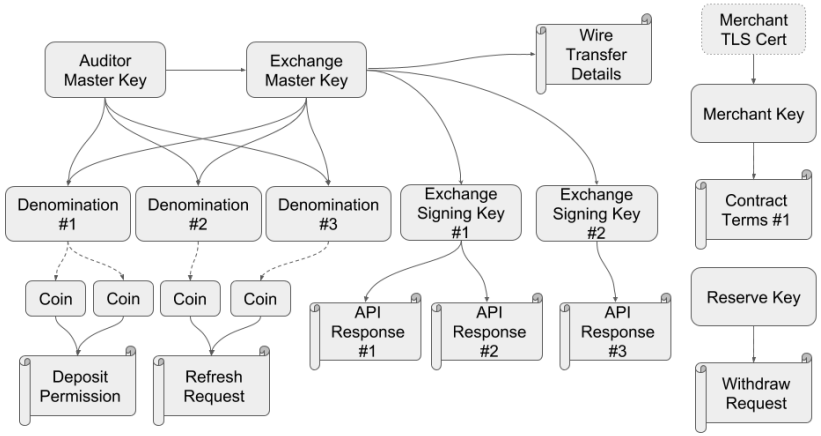


TALER

Online keys

The exchange needs RSA and EdDSA keys to be available for online signing.

- Knowledge of these private keys will allow an adversary to mint digital cash, possibly resulting in huge financial losses (eventually, this will be detected by the auditor, but only after some financial losses have been irrevocably incurred).
- The corresponding public keys are certified using Taler's public key infrastructure (which uses offline-only keys).



`taler-exchange-offline` can also be used to **revoke** the online signing keys, if we find they have been compromised.

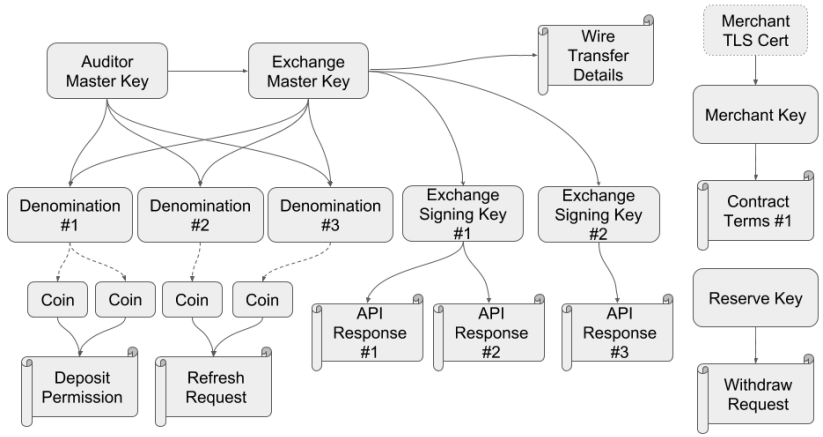
Protecting online keys

The exchange needs RSA and EdDSA keys to be available for online signing.

- `taler-exchange-secmod-rsa` and `taler-exchange-secmod-eddsa` are the only processes that must have access to the private keys.
- The secmod processes should run under a different UID, but share the same GID with the exchange.
- The secmods generate the keys, allow `taler-exchange-httpd` to sign with them, and eventually delete the private keys.
- Communication between secmods and `taler-exchange-httpd` is via a UNIX domain socket.
- Online private keys are stored on disk (not in database!) and should NOT be backed up (RAID should suffice). If disk is lost, we can always create fresh replacement keys!

Online keys

- The exchange needs d and w to be available for online signing.
- The corresponding public keys W and (e, n) are certified using Taler's public key infrastructure (which uses offline-only keys).

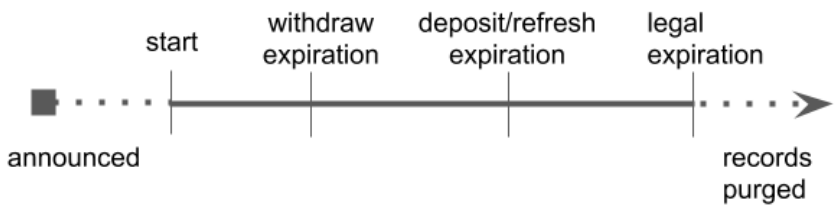


What happens if those private keys are compromised?

Denomination key (e, n) compromise

- An attacker who learns d can sign an arbitrary number of illicit coins into existence and deposit them.
- Auditor and exchange can detect this once the total number of deposits (illicit and legitimate) exceeds the number of legitimate coins the exchange created.
- At this point, (e, n) is *revoked*. Users of *unspent* legitimate coins reveal b from their withdrawal operation and obtain a *refund*.
- The financial loss of the exchange is *bounded* by the number of legitimate coins signed with d.

$\Rightarrow$ Taler frequently rotates denomination signing keys and deletes d after the signing period of the respective key expires.



TALER

Online signing key W compromise

- An attacker who learns w can sign deposit confirmations.
- Attacker sets up two (or more) merchants and customer(s) which double-spend legitimate coins at both merchants.
- The merchants only deposit each coin once at the exchange and get paid once.
- The attacker then uses w to fake deposit confirmations for the double-spent transactions.
- The attacker uses the faked deposit confirmations to complain to the auditor that the exchange did not honor the (faked) deposit confirmations.

The auditor can then detect the double-spending, but cannot tell who is to blame, and (likely) would presume an evil exchange, forcing it to pay both merchants.

Detecting online signing key W compromise

- Merchants are required to *probabilistically* report signed deposit confirmations to the auditor.
- Auditor can thus detect exchanges not reporting signed deposit confirmations.

$\Rightarrow$ Exchange can rekey if illicit key use is detected, then only has to honor deposit confirmations it already provided to the auditor *and* those without proof of double-spending *and* those merchants reported to the auditor.

$\Rightarrow$ Merchants that do not participate in reporting to the auditor risk their deposit permissions being voided in cases of an exchange's private key being compromised.

Warranting deposit safety

Exchange has *another* online signing key $W = wG$:

Sends $EdDSA_w(M, H(D), FDH(C))$ to the merchant.

This signature means that M was the *first* to deposit C and that the exchange thus must pay M.

Without this, an evil exchange could renege on the deposit confirmation and claim double-spending if a coin were deposited twice, and then not pay either merchant!

Database

The exchange needs the database to detect double spending.

- Loss of the database will allow technically skilled people to double-spend their digital cash, possibly resulting in significant financial losses.
- The database contains total amounts customers withdrew and merchants received, so sensitive private banking data. It must thus not become public.
- The auditor must have a (current) copy. Asynchronous replication should be sufficient. This copy can also serve as an additional (off-site?) backup.

taler-exchange-wirewatch

`taler-exchange-wirewatch`

needs credentials to access data about incoming wire transfers from the Nexus.

- This tool should run as a separate UID and GID (from `taler-exchange-httpd`).
- It must have access to the Postgres database (SELECT + INSERT).
- Its configuration file contains the credentials to talk to Nexus.

$\Rightarrow$ Configuration should be separate from `taler-exchange-httpd`.

taler-exchange-transfer

Only `taler-exchange-transfer` needs credentials to initiate wire transfers using the Nexus.

- This tool should run as a separate UID and GID (from `taler-exchange-httpd`).
- It must have access to the Postgres database (SELECT + INSERT).
- Its configuration file contains the credentials to talk to Nexus.

$\Rightarrow$ Configuration should be separate from `taler-exchange-httpd`.

libeufin-nexus

libeufin-nexus has to be able to interact with the escrow account of the exchange.

- It must have the private keys to sign EBICS/FinTS messages.
- It also has its own local database.
- The Nexus user and database should be kept separate from the other exchange users and the Taler exchange database.

Hardware

General notions:

- Platforms with disabled Intel ME & disabled remote administration are safer.
- VMs are not a security mechanism. Side-channel attacks abound. Avoid running any Taler component in a virtual machine “for security”.

Operating system

General notions:

- It should be safe to run the different Taler components (including Nginx, Nexus and Postgres) all on the same physical hardware (under different UIDs/GIDs). We would separate them onto different physical machines during scale-out, but not necessarily for "basic" security.
- Limiting and auditing system administrator access will be crucial.
- We recommend to **not** use any anti-virus.
- We recommend using a well-supported GNU/Linux operating system (such as Debian or Ubuntu).

Network

- We recommend to **not** use any host-based firewall. Taler components can use UNIX domain sockets (or bind to localhost).
- A network-based firewall is not required, but as long as TCP 80/443 are open Taler should work fine.
- Any firewall must be configured to permit connection to Auditor for database synchronization.
- We recommend running the Taler exchange behind an Nginx or Apache proxy for TLS termination.
- We recommend using static IP address configurations (IPv4 and IPv6).
- We recommend using DNSSEC with DANE in addition to TLS certificates.
- We recommend auditing the TLS setup using `https://observatory.mozilla.org`.

Offline payments

Requirements: Online vs. Offline Digital Currencies

https://taler.net/papers/euro-bearer-online-2021.pdf

- Offline capabilities are sometimes cited as a requirement for digital payment solutions
- All implementations must either use restrictive hardware elements and/or introduce counterparty risk.

$\Rightarrow$ Permanent offline features weaken a digital payment solution (privacy, security)

$\Rightarrow$ Introduces unwarranted competition for physical cash (endangers emergency-preparedness).

We recommend a tiered approach:

1. Online-first, bearer-based digital currency with Taler
2. (Optional:) Limited offline mode for network outages
3. Physical cash for emergencies (power outage, catastrophic cyber incidents)

Fully Offline Payments **(WiP)**

https://docs.taler.net/design-documents/030-offline-payments.html

Many central banks today demand offline capabilities for digital payment solutions.

Three possible approaches:

1. Trust-based offline payments (has counterparty and/or privacy risks)
2. Full HSM Taler wallet (has hardware costs)
3. Light-weight HSM balance register

A Scenario

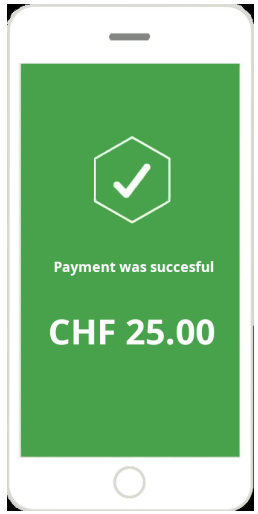
God is offline, but customer pays online

Typical Payment Process

All equivalent: Twint, PayPal, AliPay, PayTM

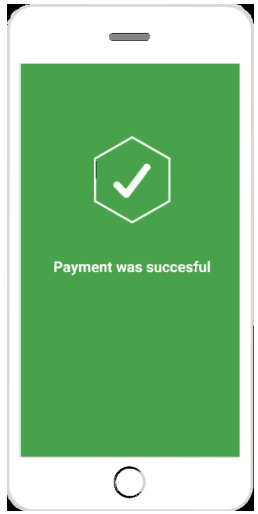
Secure Payment ...

Everything green?



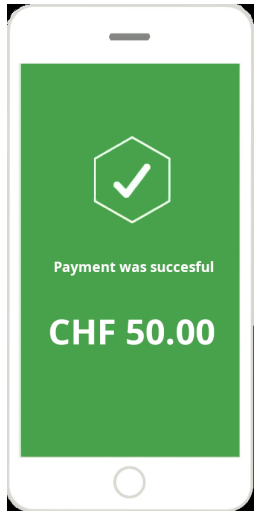
Exploit “Code”

Programming optional

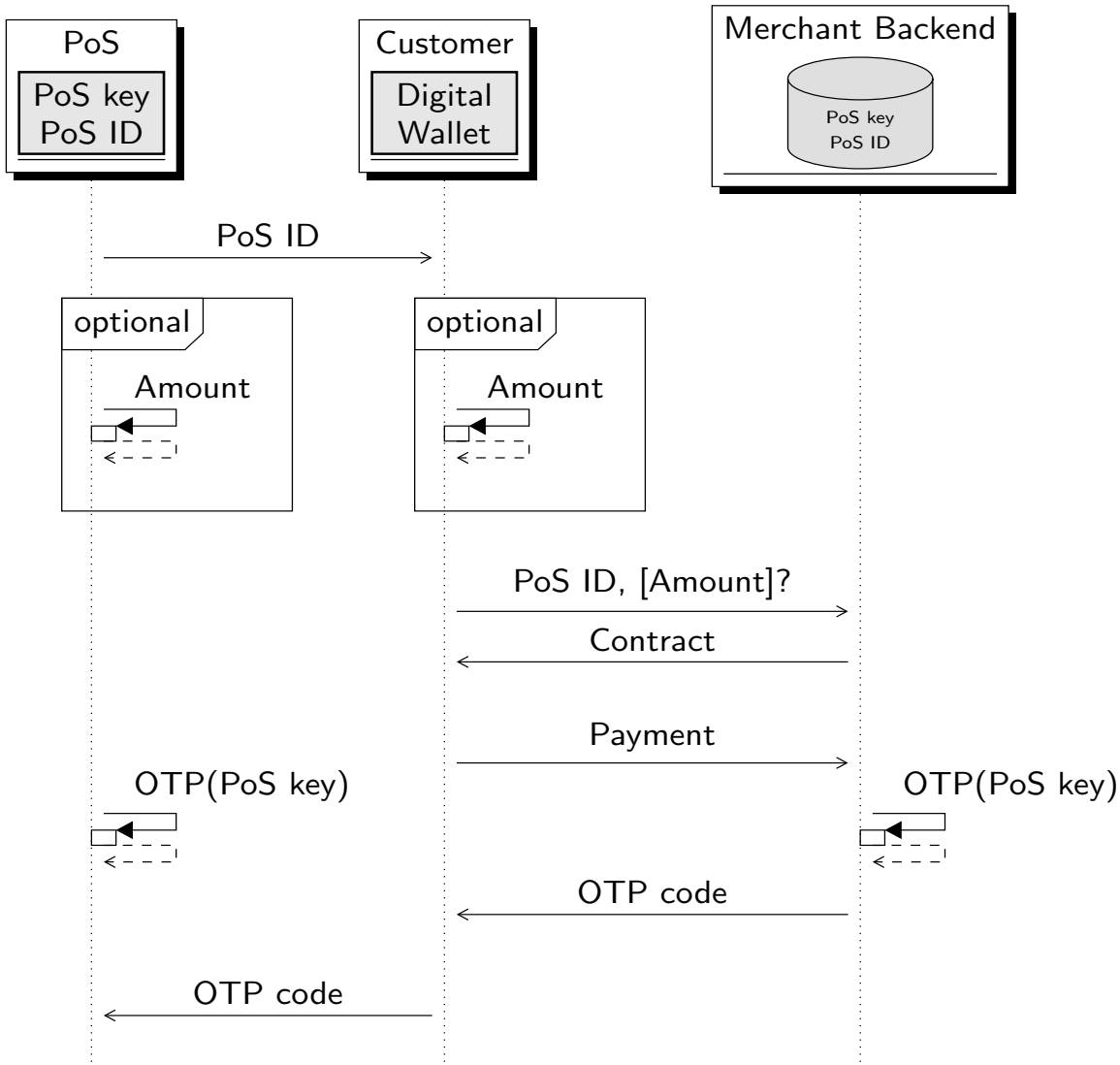


“Customers” *love* Twint ...

Daily non-business for shops



Partially Offline Payments with GNU Taler[3]



[3] Joint work with Emmanuel Benoist, Priscilla Huang and Sebastian Marchano

Programmable money: Age restrictions

TALER

Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

1. ID Verification
2. Restricted Accounts
3. Attribute-based

TALER

Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

	Privacy
1. ID Verification	bad
2. Restricted Accounts	bad
3. Attribute-based	good

TALER

Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

	Privacy	Ext. authority
1. ID Verification	bad	required
2. Restricted Accounts	bad	required
3. Attribute-based	good	required

TALER

Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

	Privacy	Ext. authority
1. ID Verification	bad	required
2. Restricted Accounts	bad	required
3. Attribute-based	good	required

Principle of Subsidiarity is violated

TALER

Principle of Subsidiarity

Functions of government—such as granting and restricting rights—should be performed *at the lowest level of authority possible*, as long as they can be performed *adequately*.

TALER

Principle of Subsidiarity

Functions of government—such as granting and restricting rights—should be performed *at the lowest level of authority possible*, as long as they can be performed *adequately*.

For age-restriction, the lowest level of authority is:

Parents, guardians and caretakers

Age restriction design for GNU Taler

Design and implementation of an age restriction scheme with the following goals:

1. It ties age restriction to the **ability to pay** (not to ID's)
2. maintains **anonymity of buyers**
3. maintains **unlinkability of transactions**
4. aligns with **principle of subsidiartiy**
5. is **practical and efficient**

TALER

Age restriction

Assumptions and scenario

- Assumption: Checking accounts are under control of eligible adults/guardians.

Age restriction

Assumptions and scenario

- Assumption: Checking accounts are under control of eligible adults/guardians.
- *Guardians* **commit** to an maximum age

Age restriction

Assumptions and scenario

- Assumption: Checking accounts are under control of eligible adults/guardians.
- *Guardians* **commit** to an maximum age
- *Minors* **attest** their adequate age

Age restriction

Assumptions and scenario

- Assumption: Checking accounts are under control of eligible adults/guardians.
- *Guardians* **commit** to an maximum age
- *Minors* **attest** their adequate age
- *Merchants* **verify** the attestations

Age restriction

Assumptions and scenario

- Assumption: Checking accounts are under control of eligible adults/guardians.
- *Guardians* **commit** to an maximum age
- *Minors* **attest** their adequate age
- *Merchants* **verify** the attestations
- Minors **derive** age commitments from existing ones

Age restriction

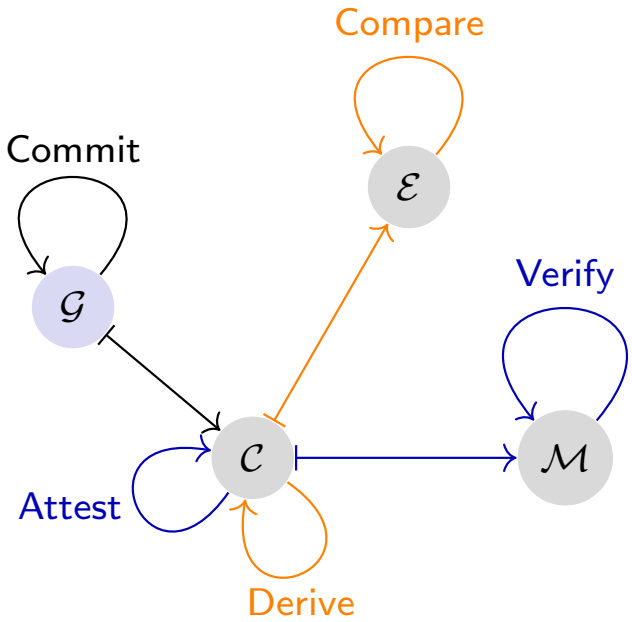
Assumptions and scenario

- Assumption: Checking accounts are under control of eligible adults/guardians.
- *Guardians* **commit** to an maximum age
- *Minors* **attest** their adequate age
- *Merchants* **verify** the attestations
- Minors **derive** age commitments from existing ones
- *Exchanges* **compare** the derived age commitments

Age restriction

Assumptions and scenario

- Assumption: Checking accounts are under control of eligible adults/guardians.
- *Guardians* **commit** to an maximum age
- *Minors* **attest** their adequate age
- *Merchants* **verify** the attestations
- Minors **derive** age commitments from existing ones
- *Exchanges* **compare** the derived age commitments



Note: Scheme is independent of payment service protocol.

Formal Function Signatures

Searching for functions

- Commit
- Attest
- Verify
- Derive
- Compare

Formal Function Signatures

Searching for functions with the following signatures

Commit :	$(a, \omega) \mapsto (Q, P)$	$\mathbb{N}_M \times \Omega \to \mathbb{O} \times \mathbb{P}$,
Attest		
Verify		
Derive		
Compare		

Mnemonics:
$\mathbb{O}$ = *c*$\mathbb{O}$*mmitments*, Q = *Q-mitment* (commitment), $\mathbb{P}$ = $\mathbb{P}$*roofs*,

TALER

Formal Function Signatures

Searching for functions with the following signatures

Commit :	$(a, \omega) \mapsto (Q, P)$	$\mathbb{N}_M \times \Omega \to \mathbb{O} \times \mathbb{P}$,
Attest :	$(m, Q, P) \mapsto T$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \to \mathbb{T} \cup \{\perp\}$,
Verify		
Derive		
Compare		

Mnemonics:
$\mathbb{O}$ = *c*$\mathbb{O}$*mmitments*, Q = *Q-mitment* (commitment), $\mathbb{P}$ = $\mathbb{P}$*roofs*, P = P*roof*,
$\mathbb{T}$ = *a*$\mathbb{T}$*testations*, T = *a*T*testation*,

TALER

Formal Function Signatures

Searching for functions with the following signatures

Commit :	$(a, \omega) \mapsto (Q, P)$	$\mathbb{N}_M \times \Omega \to \mathbb{O} \times \mathbb{P}$,
Attest :	$(m, Q, P) \mapsto T$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \to \mathbb{T} \cup \{\perp\}$,
Verify :	$(m, Q, T) \mapsto b$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{T} \to \mathbb{Z}_2$,
Derive		
Compare		

Mnemonics:
$\mathbb{O} = c\mathbb{O}mmitments$, Q = *Q-mitment* (commitment), $\mathbb{P} = \mathbb{P}roofs$, P = P*roof*,
$\mathbb{T} = a\mathbb{T}testations$, T = *a*T*testation*,

Formal Function Signatures

Searching for functions with the following signatures

Commit :	$(a, \omega) \mapsto (Q, P)$	$\mathbb{N}_M \times \Omega \to \mathbb{O} \times \mathbb{P}$,
Attest :	$(m, Q, P) \mapsto T$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \to \mathbb{T} \cup \{\perp\}$,
Verify :	$(m, Q, T) \mapsto b$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{T} \to \mathbb{Z}_2$,
Derive :	$(Q, P, \omega) \mapsto (Q', P', \beta)$	$\mathbb{O} \times \mathbb{P} \times \Omega \to \mathbb{O} \times \mathbb{P} \times \mathbb{B}$,
Compare		

Mnemonics:
$\mathbb{O} = c\mathbb{O}mmitments$, $Q = Q\text{-}mitment$ (commitment), $\mathbb{P} = \mathbb{P}roofs$, $P = Proof$,
$\mathbb{T} = a\mathbb{T}testations$, $T = aTtestation$, $\mathbb{B} = \mathbb{B}lindings$, $\beta = \beta linding$.

Formal Function Signatures

Searching for functions with the following signatures

$$
\begin{aligned}
&\text{Commit}: && (a, \omega) \mapsto (Q, P) && \mathbb{N}_M \times \Omega \to \mathbb{O} \times \mathbb{P},\\
&\text{Attest}: && (m, Q, P) \mapsto T && \mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \to \mathbb{T} \cup \{\bot\},\\
&\text{Verify}: && (m, Q, T) \mapsto b && \mathbb{N}_M \times \mathbb{O} \times \mathbb{T} \to \mathbb{Z}_2,\\
&\text{Derive}: && (Q, P, \omega) \mapsto (Q', P', \beta) && \mathbb{O} \times \mathbb{P} \times \Omega \to \mathbb{O} \times \mathbb{P} \times \mathbb{B},\\
&\text{Compare}: && (Q, Q', \beta) \mapsto b && \mathbb{O} \times \mathbb{O} \times \mathbb{B} \to \mathbb{Z}_2,
\end{aligned}
$$

Mnemonics:
$\mathbb{O} = c\mathbb{O}mmitments$, $Q = Q\text{-}mitment$ (commitment), $\mathbb{P} = \mathbb{P}roofs$, $P = Proof$,
$\mathbb{T} = a\mathbb{T}testations$, $T = aTtestation$, $\mathbb{B} = \mathbb{B}lindings$, $\beta = \beta linding$.

TALER

Formal Function Signatures

Searching for functions with the following signatures

$$
\begin{array}{lcl}
\text{Commit}: & (\mathsf{a}, \omega) \mapsto (\mathsf{Q}, \mathsf{P}) & \mathbb{N}_{\mathsf{M}} \times \Omega \to \mathbb{O} \times \mathbb{P}, \\
\text{Attest}: & (\mathsf{m}, \mathsf{Q}, \mathsf{P}) \mapsto \mathsf{T} & \mathbb{N}_{\mathsf{M}} \times \mathbb{O} \times \mathbb{P} \to \mathbb{T} \cup \{\perp\}, \\
\text{Verify}: & (\mathsf{m}, \mathsf{Q}, \mathsf{T}) \mapsto b & \mathbb{N}_{\mathsf{M}} \times \mathbb{O} \times \mathbb{T} \to \mathbb{Z}_2, \\
\text{Derive}: & (\mathsf{Q}, \mathsf{P}, \omega) \mapsto (\mathsf{Q}', \mathsf{P}', \beta) & \mathbb{O} \times \mathbb{P} \times \Omega \to \mathbb{O} \times \mathbb{P} \times \mathbb{B}, \\
\text{Compare}: & (\mathsf{Q}, \mathsf{Q}', \beta) \mapsto b & \mathbb{O} \times \mathbb{O} \times \mathbb{B} \to \mathbb{Z}_2,
\end{array}
$$

with $\Omega, \mathbb{P}, \mathbb{O}, \mathbb{T}, \mathbb{B}$ sufficiently large sets.

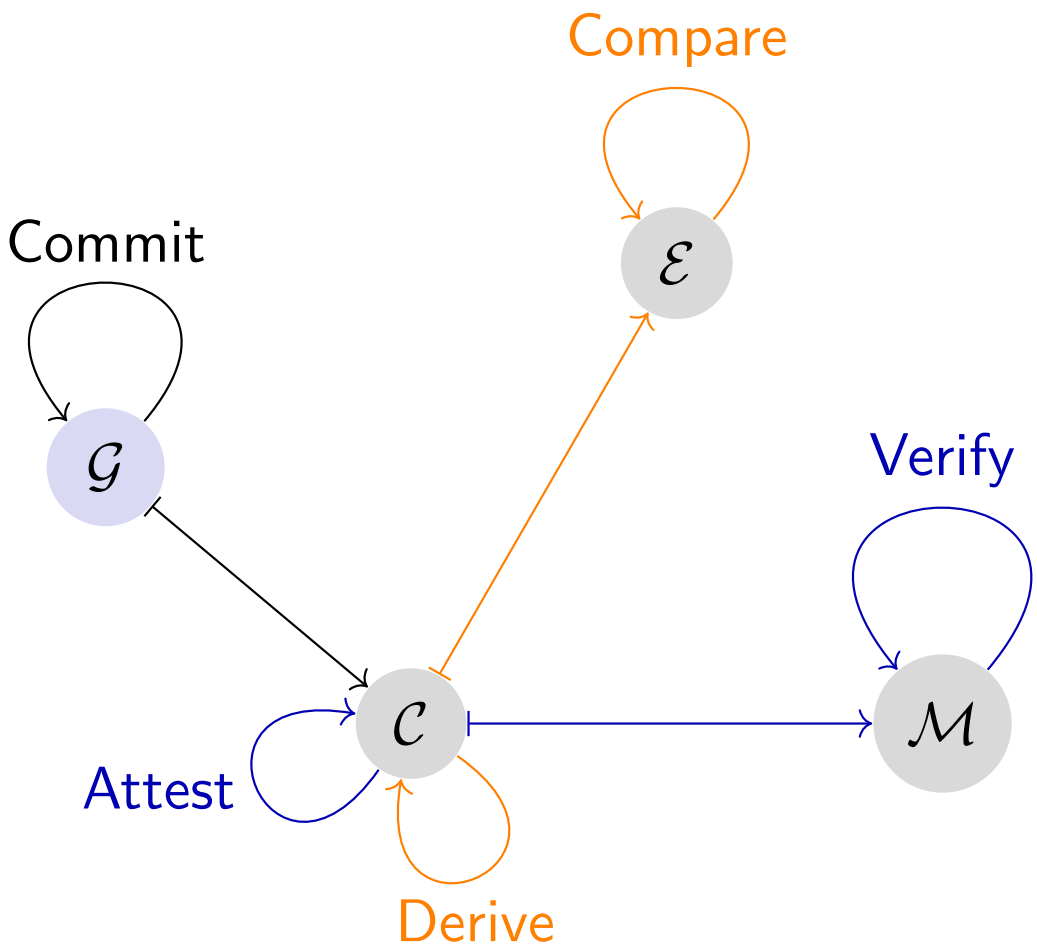
Basic and security requirements are defined later.

Mnemonics:
$\mathbb{O} = c\mathbb{O}mmitments$, $\mathsf{Q} = Q\text{-}mitment$ (commitment), $\mathbb{P} = \mathbb{P}roofs$, $\mathsf{P} = \mathsf{P}roof$,
$\mathbb{T} = a\mathbb{T}testations$, $\mathsf{T} = a\mathsf{T}testation$, $\mathbb{B} = \mathbb{B}lindings$, $\beta = \beta linding$.

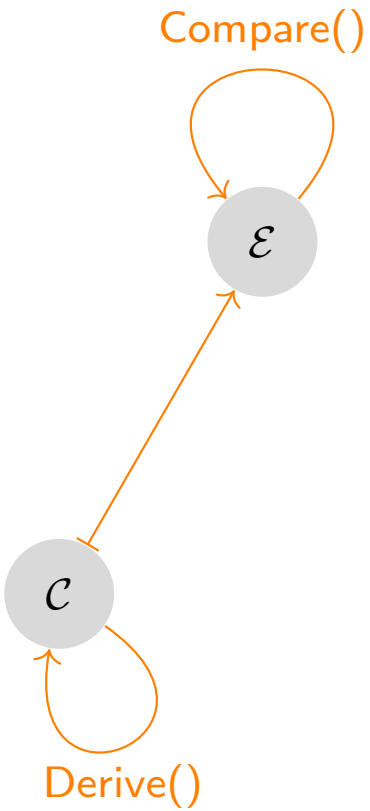
TALER

Age restriction

Naïve scheme

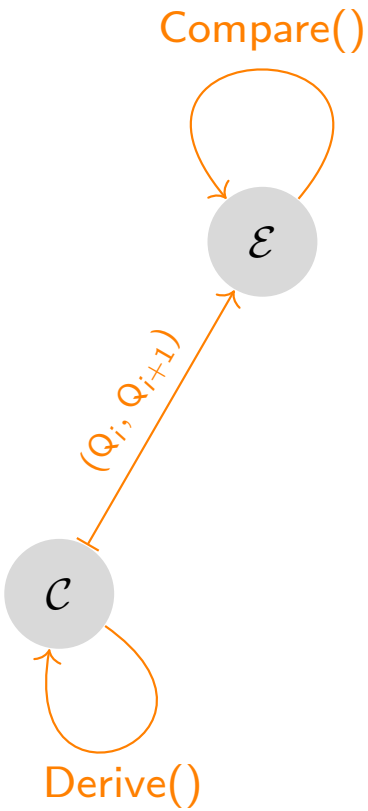


Achieving Unlinkability



Simple use of Derive() and Compare() is problematic.

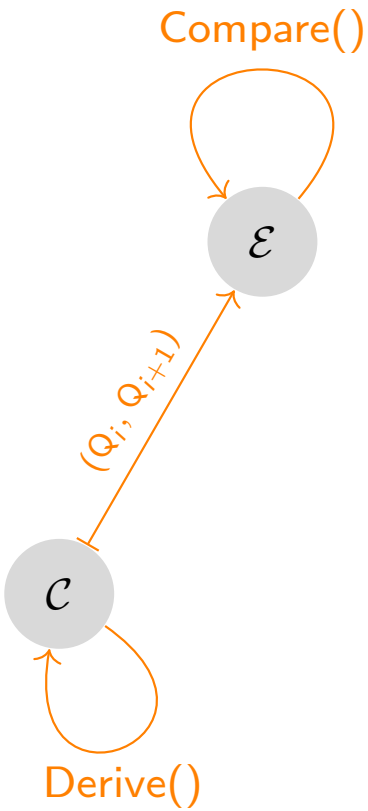
Achieving Unlinkability



Simple use of Derive() and Compare() is problematic.

- Calling Derive() iteratively generates sequence $(Q_0, Q_1, \dots)$ of commitments.
- Exchange calls Compare$(Q_i, Q_{i+1}, .)$

Achieving Unlinkability



Simple use of Derive() and Compare() is problematic.

- Calling Derive() iteratively generates sequence $(Q_0, Q_1, \dots)$ of commitments.
- Exchange calls Compare$(Q_i, Q_{i+1}, .)$

$\Longrightarrow$ **Exchange identifies sequence**

$\Longrightarrow$ **Unlinkability broken**

Achieving Unlinkability

Define cut&choose protocol DeriveCompare$_\kappa$, using Derive() and Compare().

Achieving Unlinkability

Define cut&choose protocol $\text{DeriveCompare}_\kappa$, using Derive() and Compare().

Sketch:

1. $\mathcal{C}$ derives commitments $(Q_1, \ldots, Q_\kappa)$ from Q_0 by calling Derive() with blindings $(\beta_1, \ldots, \beta_\kappa)$
2. $\mathcal{C}$ calculates $h_0 := H\left(H(Q_1, \beta_1)|| \ldots ||H(Q_\kappa, \beta_\kappa)\right)$
3. $\mathcal{C}$ sends Q_0 and h_0 to $\mathcal{E}$
4. $\mathcal{E}$ chooses $\gamma \in \{1, \ldots, \kappa\}$ randomly
5. $\mathcal{C}$ reveals $h_\gamma := H(Q_\gamma, \beta_\gamma)$ and all (Q_i, β_i), except (Q_γ, β_γ)
6. $\mathcal{E}$ compares h_0 and $H\left(H(Q_1, \beta_1)||...||h_\gamma||...||H(Q_\kappa, \beta_\kappa)\right)$ and evaluates $\text{Compare}(Q_0, Q_i, \beta_i)$.

Note: Scheme is similar to the *refresh* protocol in GNU Taler.

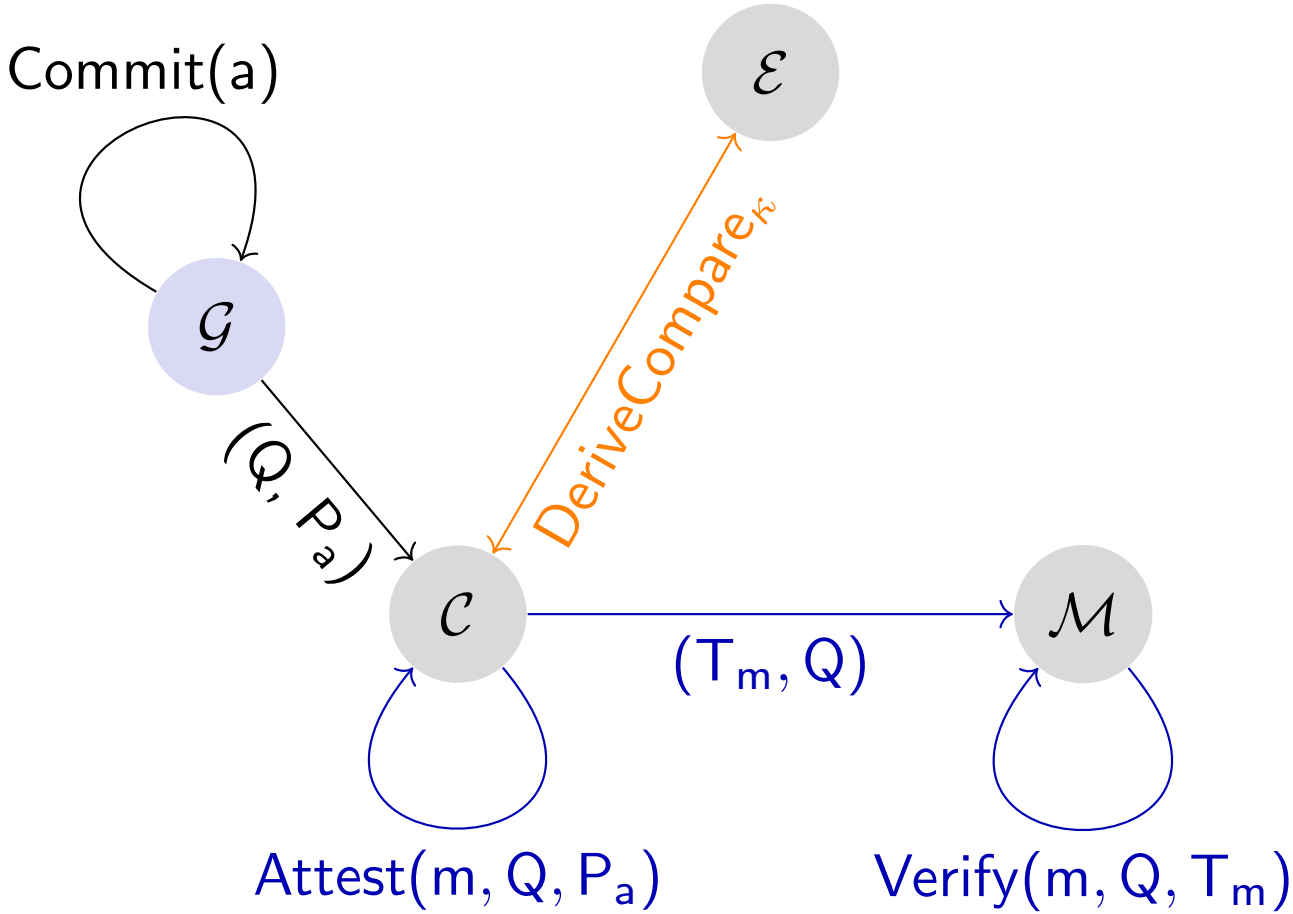
Achieving Unlinkability

With DeriveCompare$_\kappa$

- $\mathcal{E}$ learns nothing about Q_γ,
- trusts outcome with $\frac{\kappa-1}{\kappa}$ certainty,
- i.e. $\mathcal{C}$ has $\frac{1}{\kappa}$ chance to cheat.

Note: Still need Derive and Compare to be defined.

Refined scheme



Achieving Unlinkability

$\text{DeriveCompare}_\kappa : \mathbb{O} \times \mathbb{P} \times \Omega \to \{0,1\}$

$\text{DeriveCompare}_\kappa(\mathsf{Q},\mathsf{P},\omega) =$

$\mathcal{C}$:
1. *for all* $i \in \{1,\ldots,\kappa\} : (\mathsf{Q}_i,\mathsf{P}_i,\beta_i) \leftarrow \text{Derive}(\mathsf{Q},\mathsf{P},\omega+i)$
2. $h \leftarrow \mathsf{H}\big(\mathsf{H}(\mathsf{Q}_1,\beta_1) \parallel \cdots \parallel \mathsf{H}(\mathsf{Q}_\kappa,\beta_\kappa)\big)$
3. *send* (Q,h) *to* $\mathcal{E}$

$\mathcal{E}$:
5. *save* (Q,h)
6. $\gamma \xleftarrow{\$} \{1,\ldots,\kappa\}$
7. *send* γ *to* $\mathcal{C}$

$\mathcal{C}$:
8. $h'_\gamma \leftarrow \mathsf{H}(\mathsf{Q}_\gamma,\beta_\gamma)$
9. $\mathbf{E}_\gamma \leftarrow \big[(\mathsf{Q}_1,\beta_1),\ldots,(\mathsf{Q}_{\gamma-1},\beta_{\gamma-1}),\perp,(\mathsf{Q}_{\gamma+1},\beta_{\gamma+1}),\ldots,(\mathsf{Q}_\kappa,\beta_\kappa)\big]$
10. *send* $(\mathbf{E}_\gamma,h'_\gamma)$ *to* $\mathcal{E}$

$\mathcal{E}$:
11. *for all* $i \in \{1,\ldots,\kappa\} \setminus \{\gamma\} : h_i \leftarrow \mathsf{H}(\mathbf{E}_\gamma[i])$
12. *if* $h \overset{?}{\neq} \mathsf{H}(h_1\|\ldots\|h_{\gamma-1}\|h'_\gamma\|h_{\gamma+1}\|\ldots\|h_{\kappa-1})$ *return 0*
13. *for all* $i \in \{1,\ldots,\kappa\} \setminus \{\gamma\}$*: if* $0 \overset{?}{=} \text{Compare}(\mathsf{Q},\mathsf{Q}_i,\beta_i)$ *return* 0
14. *return 1*

TALER

Basic Requirements

Candidate functions

$$(\mathsf{Commit}, \mathsf{Attest}, \mathsf{Verify}, \mathsf{Derive}, \mathsf{Compare})$$

must first meet *basic* requirements:

- Existence of attestations
- Efficacy of attestations
- Derivability of commitments and attestations

Basic Requirements

Formal Details

Existence of attestations

$$\underset{\substack{a \in \mathbb{N}_M \\ \omega \in \Omega}}{\forall} : \mathsf{Commit}(a, \omega) =: (Q, P) \implies \mathsf{Attest}(m, Q, P) = \begin{cases} T \in \mathbb{T}, & \text{if } m \leq a \\ \bot & \text{otherwise} \end{cases}$$

Efficacy of attestations

$$\mathsf{Verify}(m, Q, T) = \begin{cases} 1, \text{if} \underset{P \in \mathbb{P}}{\exists} : \mathsf{Attest}(m, Q, P) = T \\ 0 \text{ otherwise} \end{cases}$$

$$\forall_{n \leq a} : \mathsf{Verify}\big(n, Q, \mathsf{Attest}(n, Q, P)\big) = 1.$$

etc.

Requirements

Details

Derivability of commitments and proofs:

Let

$$\begin{aligned} \mathsf{a} \in \mathbb{N}_{\mathsf{M}},\ \omega_0, \omega_1 \in \Omega \\ (\mathsf{Q}_0, \mathsf{P}_0) \leftarrow \mathsf{Commit}(\mathsf{a}, \omega_0), \\ (\mathsf{Q}_1, \mathsf{P}_1, \beta) \leftarrow \mathsf{Derive}(\mathsf{Q}_0, \mathsf{P}_0, \omega_1). \end{aligned}$$

We require

$$\mathsf{Compare}(\mathsf{Q}_0, \mathsf{Q}_1, \beta) = 1$$

and for all $n \leq \mathsf{a}$:

$$\mathsf{Verify}(n, \mathsf{Q}_1, \mathsf{Attest}(n, \mathsf{Q}_1, \mathsf{P}_1)) = \mathsf{Verify}(n, \mathsf{Q}_0, \mathsf{Attest}(n, \mathsf{Q}_0, \mathsf{P}_0))$$

Security Requirements

Candidate functions must also meet *security* requirements. Those are defined via security games:

- Game: Age disclosure by commitment or attestation

$\leftrightarrow$ Requirement: Non-disclosure of age

- Game: Forging attestation

$\leftrightarrow$ Requirement: Unforgeability of minimum age

- Game: Distinguishing derived commitments and attestations

$\leftrightarrow$ Requirement: Unlinkability of commitments and attestations

Meeting the security requirements means that adversaries can win those games only with negligible advantage.

Adversaries are arbitrary polynomial-time algorithms, acting on all relevant input.

Security Requirements

Simplified Example

Game $G_{\mathcal{A}}^{\mathsf{FA}}(\lambda)$—Forging an attest:

1. $(\mathsf{a}, \omega) \xleftarrow{\$} \mathbb{N}_{\mathsf{M}-1} \times \Omega$
2. $(\mathsf{Q}, \mathsf{P}) \leftarrow \mathsf{Commit}(\mathsf{a}, \omega)$
3. $(\mathsf{m}, \mathsf{T}) \leftarrow \mathcal{A}(\mathsf{a}, \mathsf{Q}, \mathsf{P})$
4. Return 0 if $\mathsf{m} \leq \mathsf{a}$
5. Return $\mathsf{Verify}(\mathsf{m}, \mathsf{Q}, \mathsf{T})$

Requirement: Unforgeability of minimum age

$$\bigvee_{\mathcal{A} \in \mathfrak{A}(\mathbb{N}_{\mathsf{M}} \times \mathbb{O} \times \mathbb{P} \to \mathbb{N}_{\mathsf{M}} \times \mathbb{T})} : \Pr\left[G_{\mathcal{A}}^{\mathsf{FA}}(\lambda) = 1\right] \leq \epsilon(\lambda)$$

TALER

Solution: Instantiation with ECDSA

To Commit to age (group) $a \in \{1, \ldots, M\}$

Solution: Instantiation with ECDSA

To Commit to age (group) $a \in \{1, \ldots, M\}$

1. Guardian generates ECDSA-keypairs, one per age (group):

$$\langle (q_1, p_1), \ldots, (q_M, p_M) \rangle$$

Solution: Instantiation with ECDSA

To Commit to age (group) $a \in \{1, \ldots, M\}$

1. Guardian generates ECDSA-keypairs, one per age (group):

$$\langle (q_1, p_1), \ldots, (q_M, p_M) \rangle$$

2. Guardian then **drops** all private keys p_i for $i > a$:

$$\left\langle (q_1, p_1), \ldots, (q_a, p_a), (q_{a+1}, \perp), \ldots, (q_M, \perp) \right\rangle$$

- $\vec{Q} := (q_1, \ldots, q_M)$ is the *Commitment*,
- $\vec{P}_a := (p_1, \ldots, p_a, \perp, \ldots, \perp)$ is the *Proof*

TALER

Solution: Instantiation with ECDSA

To Commit to age (group) $a \in \{1, \ldots, M\}$

1. Guardian generates ECDSA-keypairs, one per age (group):

$$\langle (q_1, p_1), \ldots, (q_M, p_M) \rangle$$

2. Guardian then **drops** all private keys p_i for $i > a$:

$$\left\langle (q_1, p_1), \ldots, (q_a, p_a), (q_{a+1}, \perp), \ldots, (q_M, \perp) \right\rangle$$

   - $\vec{Q} := (q_1, \ldots, q_M)$ is the *Commitment*,
   - $\vec{P}_a := (p_1, \ldots, p_a, \perp, \ldots, \perp)$ is the *Proof*

3. Guardian gives child $\langle \vec{Q}, \vec{P}_a \rangle$

TALER

Instantiation with ECDSA

Definitions of Attest and Verify

Child has

- ordered public-keys $\vec{\mathrm{Q}} = (q_1, \ldots, q_{\mathrm{M}})$,
- (some) private-keys $\vec{\mathrm{P}} = (p_1, \ldots, p_{\mathrm{a}}, \perp, \ldots, \perp)$.

Instantiation with ECDSA

Definitions of Attest and Verify

Child has

- ordered public-keys $\vec{Q} = (q_1, \ldots, q_M)$,
- (some) private-keys $\vec{P} = (p_1, \ldots, p_a, \perp, \ldots, \perp)$.

To Attest a minimum age $m \leq a$:

Sign a message with ECDSA using private key p_m

Instantiation with ECDSA

Definitions of Attest and Verify

Child has

- ordered public-keys $\vec{Q} = (q_1, \ldots, q_M)$,
- (some) private-keys $\vec{P} = (p_1, \ldots, p_a, \perp, \ldots, \perp)$.

To Attest a minimum age $m \leq a$:

Sign a message with ECDSA using private key p_m

Merchant gets

- ordered public-keys $\vec{Q} = (q_1, \ldots, q_M)$
- Signature σ

Instantiation with ECDSA

Definitions of Attest and Verify

Child has

- ordered public-keys $\vec{Q} = (q_1, \ldots, q_M)$,
- (some) private-keys $\vec{P} = (p_1, \ldots, p_a, \perp, \ldots, \perp)$.

To Attest a minimum age $m \leq a$:

Sign a message with ECDSA using private key p_m

Merchant gets

- ordered public-keys $\vec{Q} = (q_1, \ldots, q_M)$
- Signature σ

To Verify a minimum age m:

Verify the ECDSA-Signature σ with public key q_m.

Instantiation with ECDSA

Definitions of Derive and Compare

Child has $\vec{Q} = (q_1, \ldots, q_M)$ and $\vec{P} = (p_1, \ldots, p_a, \perp, \ldots, \perp)$.

Instantiation with ECDSA

Definitions of Derive and Compare

Child has $\vec{\mathrm{Q}} = (q_1, \ldots, q_\mathrm{M})$ and $\vec{\mathrm{P}} = (p_1, \ldots, p_\mathrm{a}, \perp, \ldots, \perp)$.

To Derive new $\vec{\mathrm{Q}}'$ and $\vec{\mathrm{P}}'$: Choose random $\beta \in \mathbb{Z}_g$ and calculate

$$\vec{\mathrm{Q}}' := \big(\beta * q_1, \ldots, \beta * q_\mathrm{M}\big),$$
$$\vec{\mathrm{P}}' := \big(\beta p_1, \ldots, \beta p_\mathrm{a}, \perp, \ldots, \perp\ \big)$$

Note: $(\beta p_i) * G = \beta * (p_i * G) = \beta * q_i$

$\beta * q_i$ is scalar multiplication on the elliptic curve.

Instantiation with ECDSA

Definitions of Derive and Compare

Child has $\vec{\mathrm{Q}} = (q_1, \ldots, q_\mathrm{M})$ and $\vec{\mathrm{P}} = (p_1, \ldots, p_\mathrm{a}, \perp, \ldots, \perp)$.

To Derive new $\vec{\mathrm{Q}}'$ and $\vec{\mathrm{P}}'$: Choose random $\beta \in \mathbb{Z}_g$ and calculate

$$\vec{\mathrm{Q}}' := \big(\beta * q_1, \ldots, \beta * q_\mathrm{M}\big),$$
$$\vec{\mathrm{P}}' := \big(\beta p_1, \ldots, \beta p_\mathrm{a}, \perp, \ldots, \perp\ \big)$$

Note: $(\beta p_i) * G = \beta * (p_i * G) = \beta * q_i$

$\beta * q_i$ is scalar multiplication on the elliptic curve.

Exchange gets $\vec{\mathrm{Q}} = (q_1, \ldots, q_\mathrm{M})$, $\vec{\mathrm{Q}}' = (q'_1, \ldots, q'_\mathrm{M})$ and β

To Compare, calculate: $(\beta * q_1, \ldots, \beta * q_\mathrm{M}) \stackrel{?}{=} (q'_1, \ldots, q'_\mathrm{M})$

Instantiation with ECDSA

Functions (Commit, Attest, Verify, Derive, Compare)
as defined in the instantiation with ECDSA

- meet the basic requirements,
- also meet all security requirements.
  Proofs by security reduction, details are in the paper.

TALER

Instantiation with ECDSA

Full definitions

$$\mathsf{Commit}_{E,[\cdot]_g}(\mathsf{a},\omega) := \Big\langle \overbrace{(q_1,\ldots,q_\mathsf{M})}^{=\vec{\mathsf{Q}}}, \overbrace{(p_1,\ldots,p_\mathsf{a},\bot,\ldots,\bot)}^{=\vec{\mathsf{P}},\ \text{length M}} \Big\rangle$$

$$\mathsf{Attest}_{E,\mathsf{H}}(\mathsf{b},\vec{\mathsf{Q}},\vec{\mathsf{P}}) := \begin{cases} \mathsf{T}_\mathsf{b} := \mathsf{Sig}_{E,\mathsf{H}}\big(\mathsf{b},\vec{\mathsf{P}}[\mathsf{b}]\big) & \text{if } \vec{\mathsf{P}}[\mathsf{b}] \overset{?}{\neq} \bot \\ \bot & \text{otherwise} \end{cases}$$

$$\mathsf{Verify}_{E,\mathsf{H}}(\mathsf{b},\vec{\mathsf{Q}},\mathsf{T}) := \mathsf{Ver}_{E,\mathsf{H}}(\mathsf{b},\vec{\mathsf{Q}}[\mathsf{b}],\mathsf{T})$$

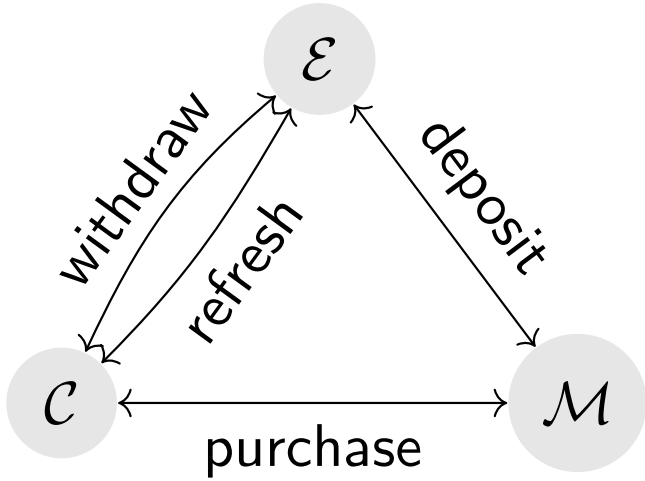
$$\mathsf{Derive}_{E,[\cdot]_g}(\vec{\mathsf{Q}},\vec{\mathsf{P}},\omega) := \Big\langle (\beta * q_1,\ldots,\beta * q_\mathsf{M}), (\beta p_1,\ldots,\beta p_\mathsf{a},\bot,\ldots,\bot), \beta \Big\rangle$$

$$\text{with } \beta := [\omega]_g \text{ and multiplication } \beta p_i \text{ modulo } g$$

$$\mathsf{Compare}_E(\vec{\mathsf{Q}},\vec{\mathsf{Q}}',\beta) := \begin{cases} 1 & \text{if } (\beta * q_1,\ldots,\beta * q_\mathsf{M}) \overset{?}{=} (q'_1,\ldots,q'_\mathsf{M}) \\ 0 & \text{otherwise} \end{cases}$$

TALER

Reminder: GNU Taler Fundamentals



- Coins are public-/private key-pairs (C_p, c_s).
- Exchange blindly signs $\mathrm{FDH}(C_p)$ with denomination key d_p
- Verification:

$$1 \stackrel{?}{=} \mathrm{SigCheck}\big(\mathrm{FDH}(C_p), D_p, \sigma_p\big)$$

(D_p = public key of denomination and σ_p = signature)

Integration with GNU Taler

Binding age restriction to coins

To bind an age commitment Q to a coin C_p, instead of signing $\mathrm{FDH}(C_p)$, $\mathcal{E}$ now blindly signs

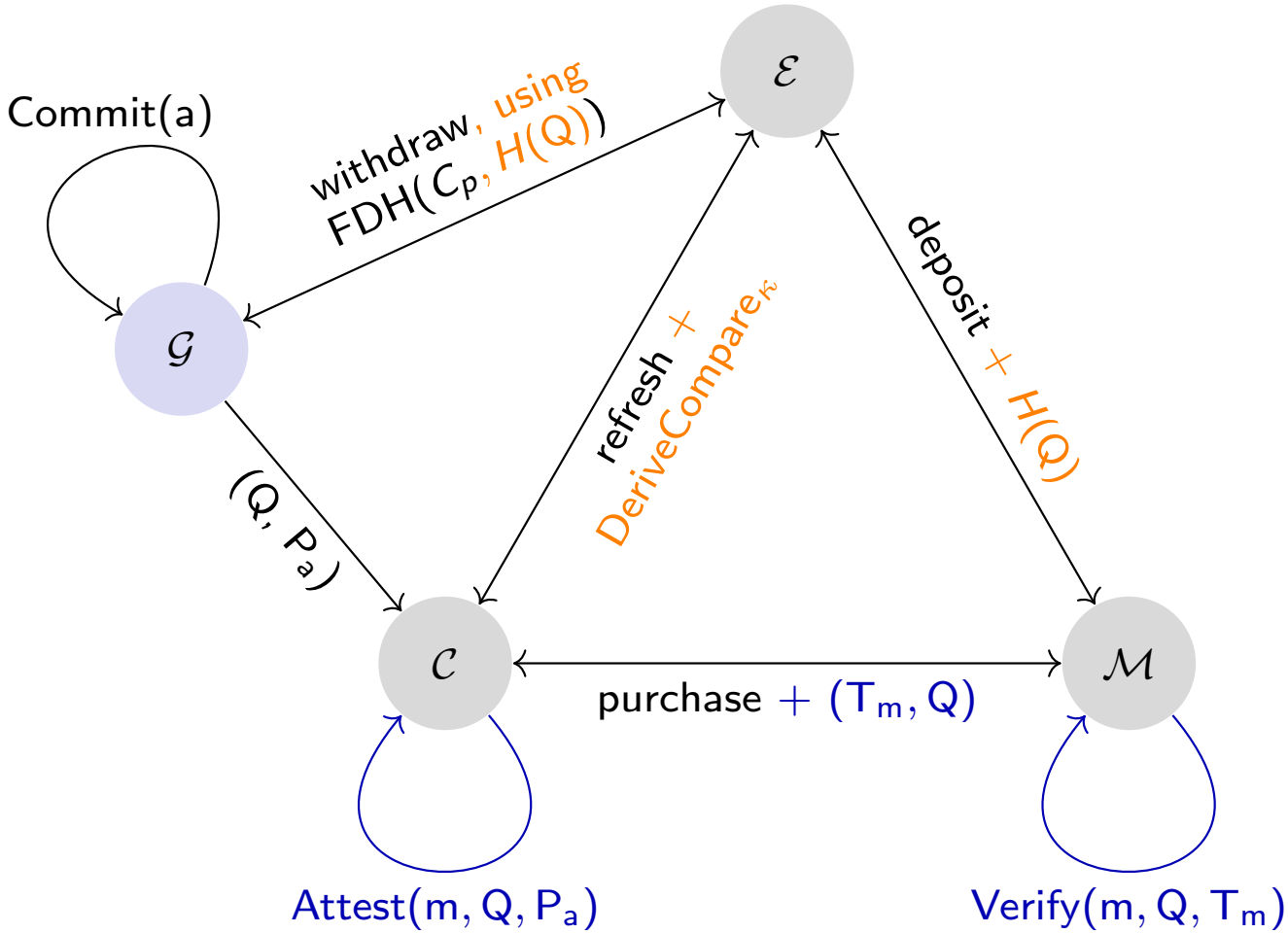
$$\mathrm{FDH}(C_p, H(\mathrm{Q}))$$

Verfication of a coin now requires $H(\mathrm{Q})$, too:

$$1 \stackrel{?}{=} \mathrm{SigCheck}\big(\mathrm{FDH}(C_p, H(\mathrm{Q})), D_p, \sigma_p\big)$$

Integration with GNU Taler

Integrated schemes



Instantiation with Edx25519

Paper also formally defines another signature scheme: Edx25519.

- Scheme already in use in GNUnet,
- based on EdDSA (Bernstein et al.),
- generates compatible signatures and
- allows for key derivation from both, private and public keys, independently.

Current implementation of age restriction in GNU Taler uses Edx25519.

Age Restrictions based on KYC

Subsidiarity requires bank accounts being owned by adults.

- Scheme can be adapted to case where minors have bank accounts
  - Assumption: banks provide minimum age information during bank transactions.
  - Child and Exchange execute a variant of the cut&choose protocol.

Discussion

- Our solution can in principle be used with any token-based payment scheme
- GNU Taler best aligned with our design goals (security, privacy and efficiency)
- Subsidiarity requires bank accounts being owned by adults
  - Scheme can be adapted to case where minors have bank accounts
    - Assumption: banks provide minimum age information during bank transactions.
    - Child and Exchange execute a variant of the cut&choose protocol.
- Our scheme offers an alternative to identity management systems (IMS)

Related Work

- Current privacy-perserving systems all based on attribute-based credentials (Koning et al., Schanzenbach et al., Camenisch et al., Au et al.)
- Attribute-based approach lacks support:
  - Complex for consumers and retailers
  - Requires trusted third authority

- Other approaches tie age-restriction to ability to pay ("debit cards for kids")
  - Advantage: mandatory to payment process
  - Not privacy friendly

Conclusion

Age restriction is a technical, ethical and legal challenge.
Existing solutions are

- without strong protection of privacy or
- based on identity management systems (IMS)

Our scheme offers a solution that is

- based on subsidiarity
- privacy preserving
- efficient
- an alternative to IMS

TALER

Software development & deployment

Development Infrastructure

- Borg: incremental backup
- Buildbot: CI/CD (https://buildbot.taler.net/)
- Davical: Caldav group calendar
- Docker: virtualization, packaging
- Git/Gitolite: distributed version control (https://git.taler.net/)
- Mailman: public e-mail lists (taler@gnu.org/)
- Mantis: bug tracker (https://bugs.taler.net/)
- Mattermost: messaging, process management (https://mattermost.taler.net/)
- Sphinx: documentation generation (HTML, PDF, info, man) (https://docs.taler.net/)
- Weblate: collaborative AI-supported internationalization (https://weblate.taler.net/)

TALER

Development Tools

- Coverity: static analysis (C/C++) (https://scan.coverity.com/)
- GNU recutils: constant registration (https://gana.gnunet.org/)
- Twister: fault injection
- Valgrind: dynamic analysis (C/C++)
- zzuf: fuzzing

Cryptographic dependencies

- libargon2
- libgcrypt
- libsodium

TALER

Additional dependencies

- libsqlite3
- libpq / Postgres
- libjansson
- libcurl
- libunistring
- **GNU libmicrohttpd**
- **GNUnet**

TALER

High-level Deployment Recipe

. . . as a bank

1. Create an escrow bank account for the exchange with EBICS access
2. Provision offline signing machine
3. Provision two PostgreSQL databases (for libeufin-nexus and exchange)
4. Provision user-facing exchange service and secmod processes
5. Provision libeufin-nexus (connected to escrow account and providing an internal API to the exchange)
6. Test using the “taler-wallet-cli“

Exchange escrow account access

The Taler exchange needs to communicate with a core banking system . . .

- to query for transactions into the exchange's escrow account
- to initiate payments of aggregated Taler deposits to merchants

In a Taler deployment, the *Taler Wire Gateway* provides an API to the exchange for Taler-specific access to the Exchange's escrow account. Multiple implementations of the Taler Wire Gateway exist:

- libeufin-bank, a self-contained play money demo bank
- libeufin-nexus, an adapter to EBICS and other protocols
- Depolymerizer, an adapter to blockchains

libeufin-nexus setup overview

https://docs.taler.net/libeufin/

- Obtain EBICS subscriber configuration (host URL, host ID, user ID, partner ID) for the bank account
- Create and back up the key material for the bank connection (contains EBICS subscriber configuration and private keys)
- Export key letter and activate subscriber in the EBICS host (physical mail)
- Confirm connection is active
- Set up scheduled tasks for ingesting new transactions / sending payment initiations

libeufin-nexus limitations at GLS Bank

The GLS accounts with EBICS access that we have access to have some limitations:

- SEPA instant credit transfers are not supported yet (by the bank)
- Erroneous payment initiations are accepted by the GLS EBICS host, but an error message is later sent only by paper mail (and not reported by the CRZ download request)
- Limited access to transaction history (3 months)

Performance[4]

[4] Joint work with Marco Boss

Performance

Other Payment Systems

Bitcoin

? TPS

[06.22] - Researchgate

Performance

Other Payment Systems

Bitcoin

4 TPS

[06.22] - Researchgate

Performance

Other Payment Systems

Bitcoin

4 TPS

[06.22] - Researchgate

Performance

Other Payment Systems

Bitcoin	PayPal
4 TPS	193 TPS

[06.22] - Researchgate

Performance

Other Payment Systems

Bitcoin

4 TPS

PayPal

193 TPS

[06.22] - Researchgate

Performance

Other Payment Systems

Bitcoin	PayPal	Visa
4 TPS	193 TPS	1'667 TPS

[06.22] - Researchgate

Performance

Other Payment Systems

Bitcoin	PayPal	Visa
4 TPS	193 TPS	1'667 TPS

[06.22] - Researchgate

Performance

CBDC Projects

e-Krona (Sweden)

100 TPS

[06.22] - Bostonfed - Atlatic Council - Riksbank

Performance

CBDC Projects

e-Krona (Sweden)

100 TPS

[06.22] - Bostonfed - Atlatic Council - Riksbank

TALER

Performance

CBDC Projects

e-Krona (Sweden)

100 TPS

e-CNY (China)

10'000 TPS

[06.22] - Bostonfed - Atlatic Council - Riksbank

Performance

CBDC Projects

e-Krona (Sweden)

100 TPS

e-CNY (China)

10'000 TPS

[06.22] - Bostonfed - Atlatic Council - Riksbank

Performance

CBDC Projects

e-Krona (Sweden)

100 TPS

e-CNY (China)

10'000 TPS

Project Hamilton (MIT)

1'700'000 TPS

[06.22] - Bostonfed - Atlatic Council - Riksbank

Performance

CBDC Projects

e-Krona (Sweden)

100 TPS

e-CNY (China)

10'000 TPS

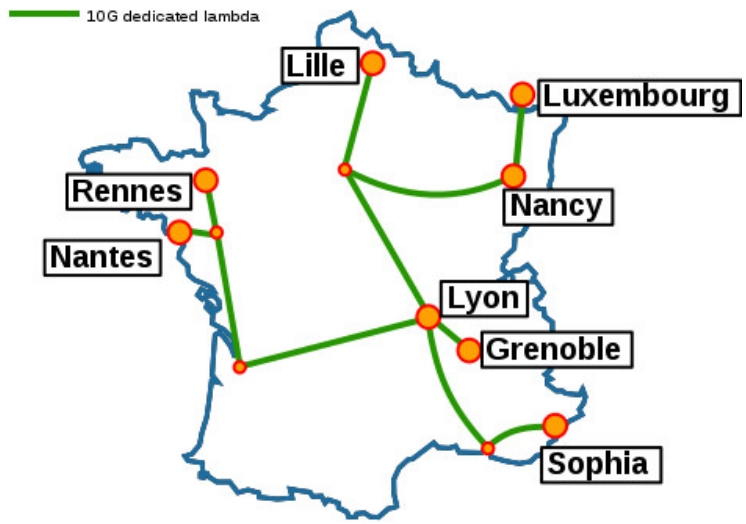
Project Hamilton (MIT)

1'700'000 TPS

[06.22] - Bostonfed - Atlatic Council - Riksbank

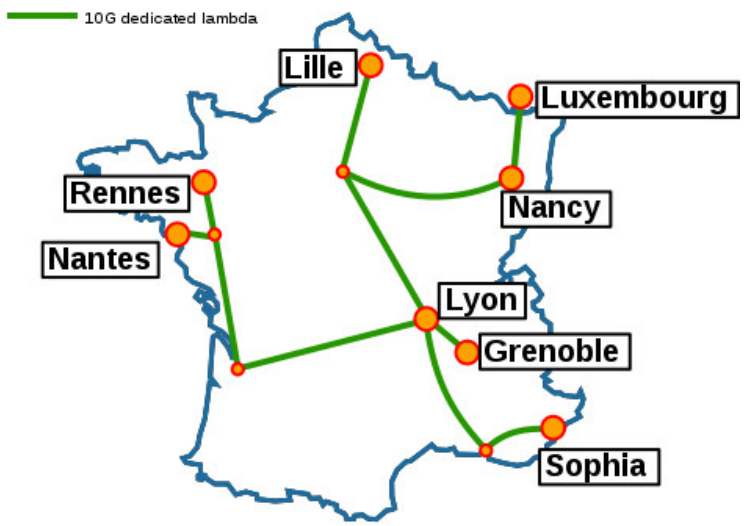
Grid'5000

- Large-scale flexible testbed



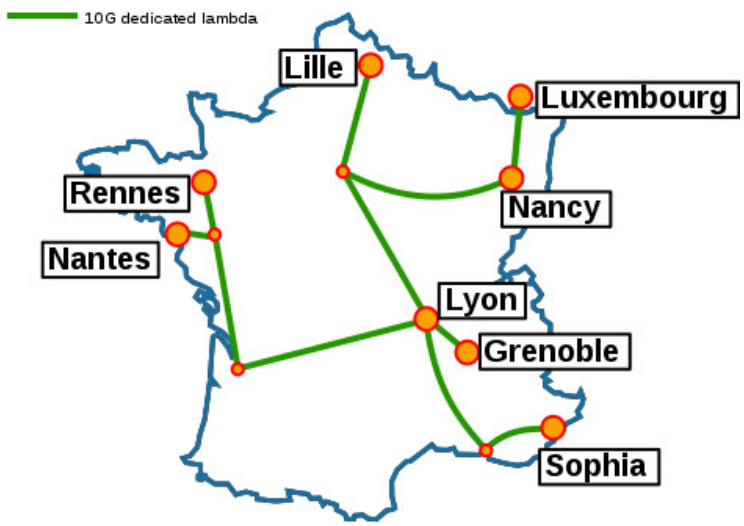
Grid'5000

- Large-scale flexible testbed
- 800 nodes with total 15'000 cores



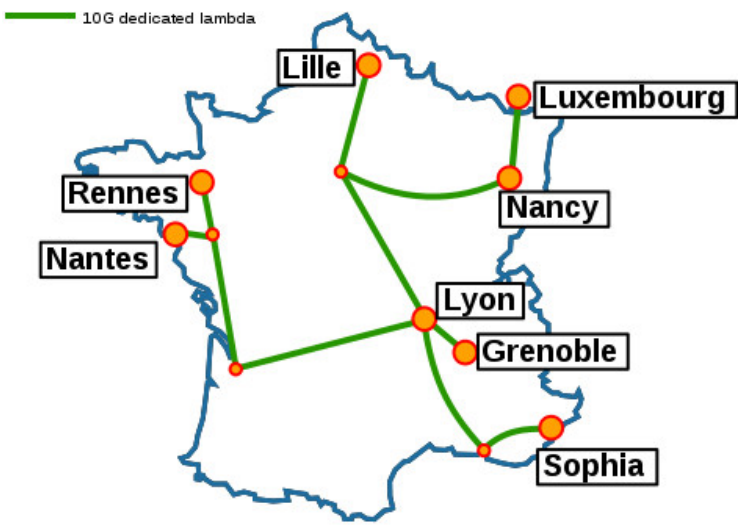
Grid'5000

- Large-scale flexible testbed
- 800 nodes with total 15'000 cores
- Bare metal deployments



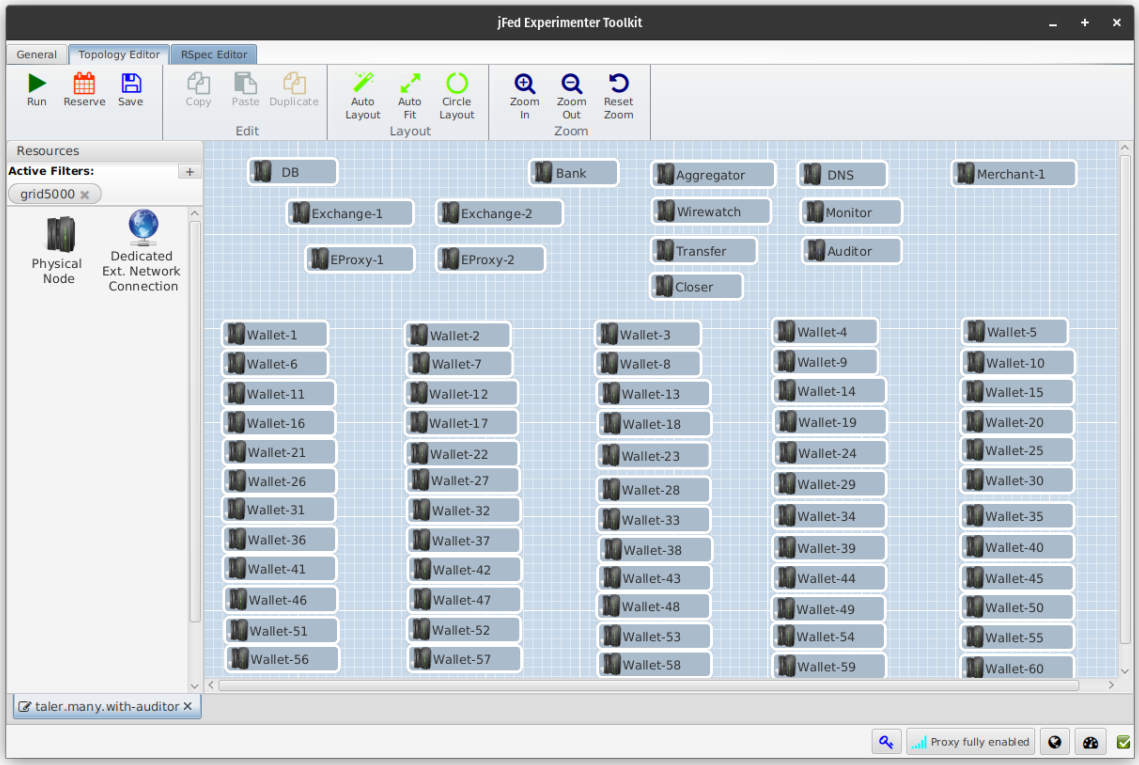
Grid'5000

- Large-scale flexible testbed
- 800 nodes with total 15'000 cores
- Bare metal deployments
- Fully customizable software stack

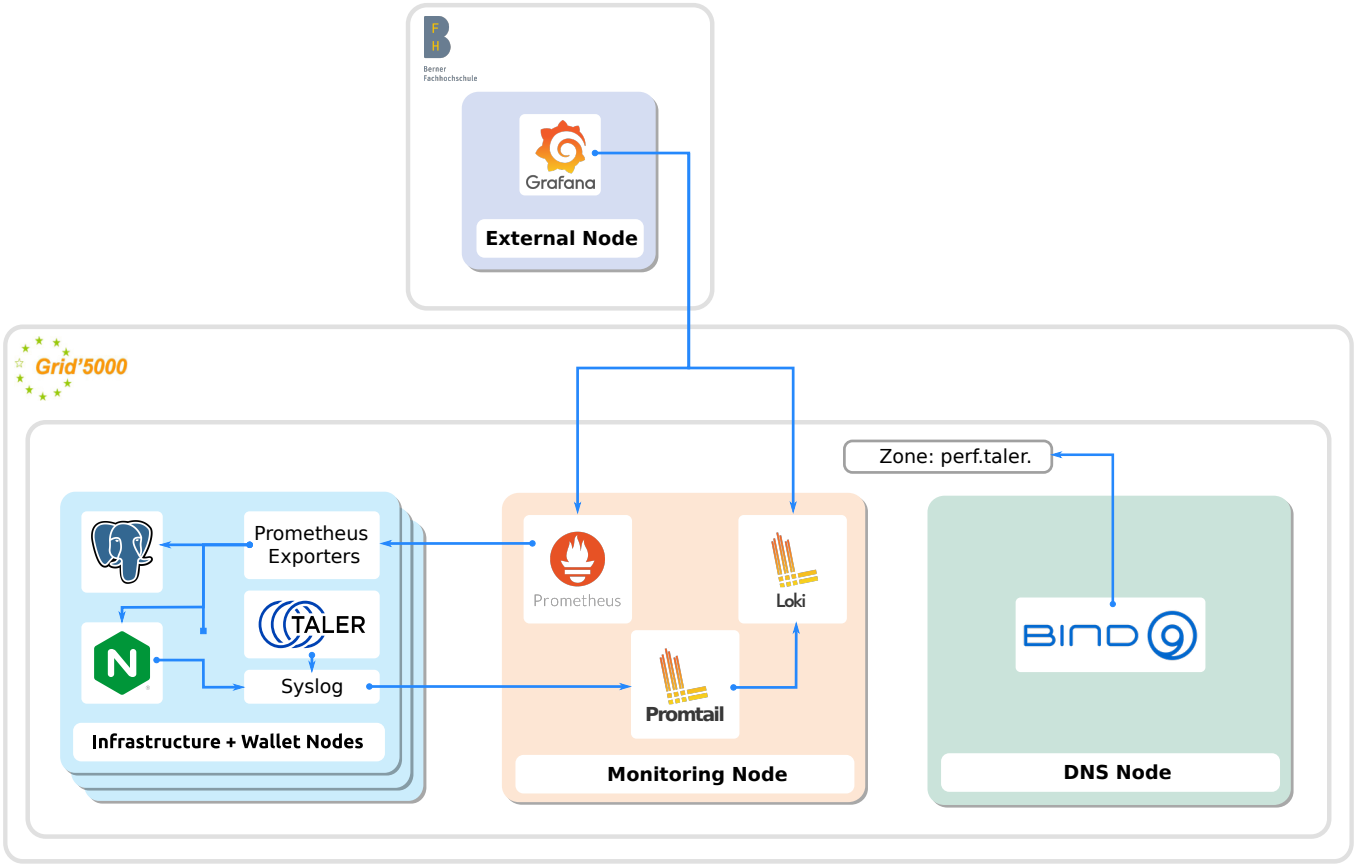


Platform Access

jFed - Java Based GUI and CLI



Architecture



Allocate an Experiment

1.

Build Image (Kameleon)

Allocate an Experiment

1.

Build Image (Kameleon)

2.

Grid'5000

Copy Image to Grid'5000

Allocate an Experiment

1.

Build Image (Kameleon)

2.

Grid'5000

Copy Image to Grid'5000

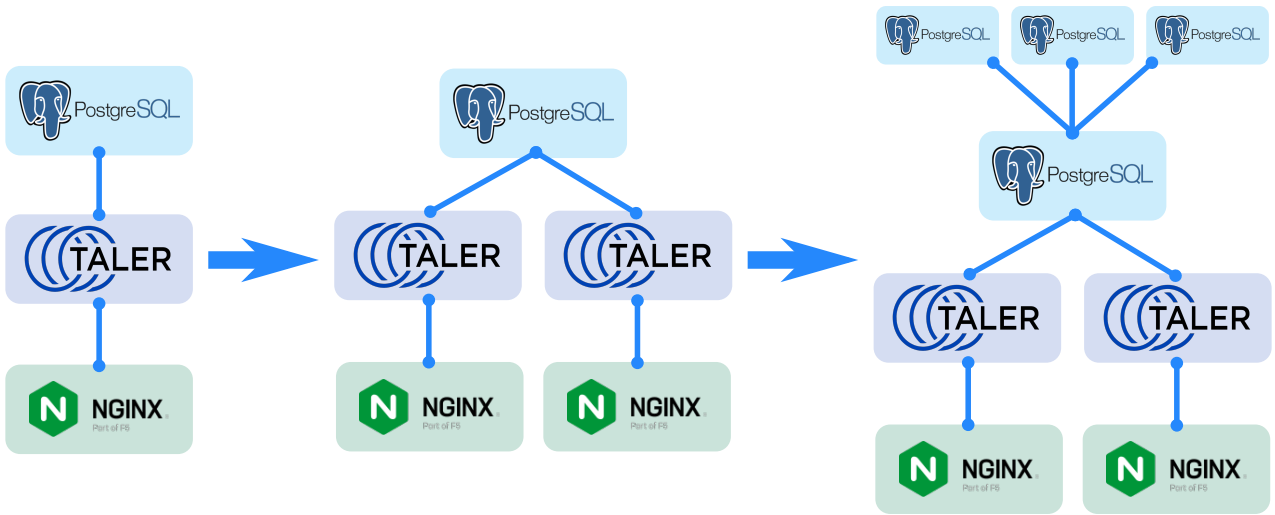
3.

Allocate Experiment (jFed)

TALER

Allocate an Experiment

Horizontal Distribution



Dashboard

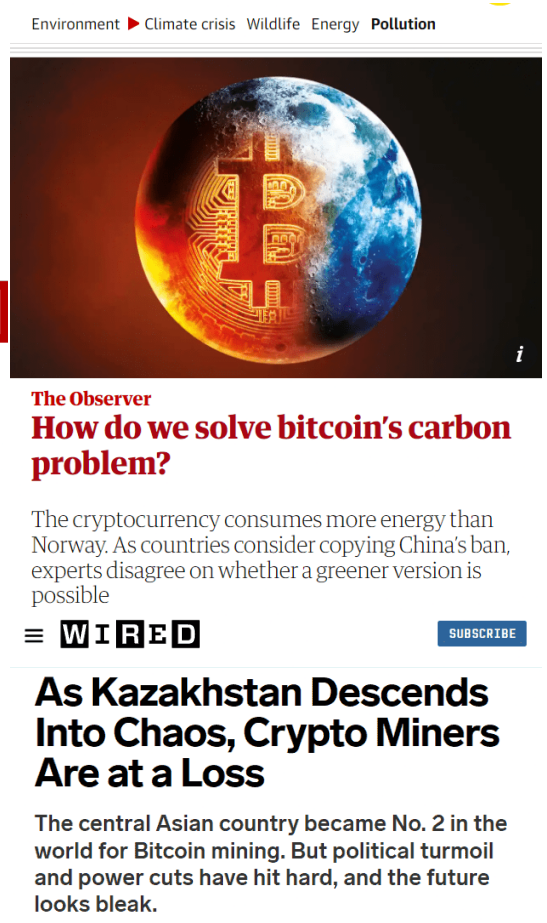
Blockchain integration: Project Depolymerization

TALER

Blockchain based cryptocurrencies

Biggest cryptocurrencies

- **BTC** Bitcoin
- **ETH** Ethereum



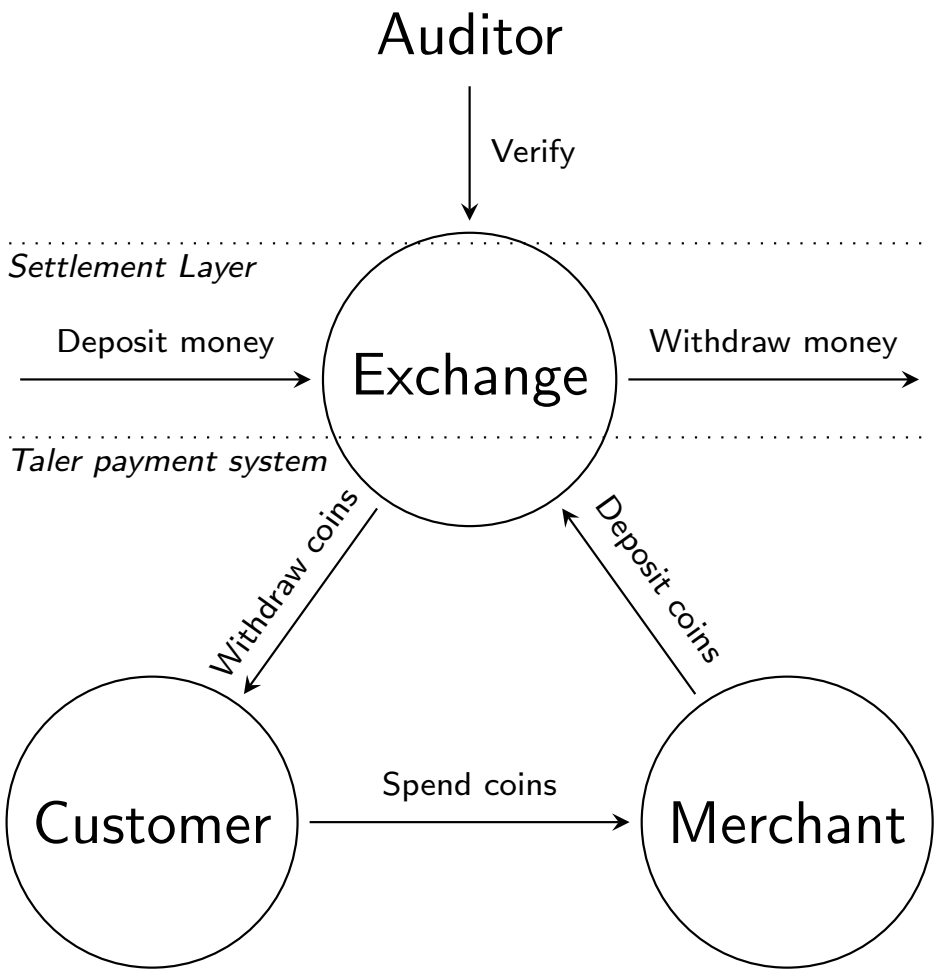
Common blockchain limitations

- **Delay** block and confirmation delay
- **Cost** transaction fees
- **Scalability** limited amount of transaction per second
- **Ecological impact** computation redundancy
- **Privacy**
- **Regulatory risk**

TALER

Taler

Architecture



Settlement layer

- For Depolymerization: Blockchain!

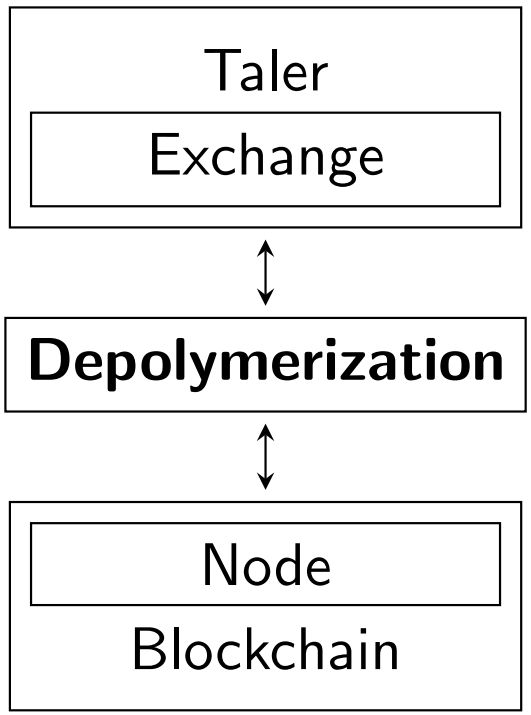
Taler payment system

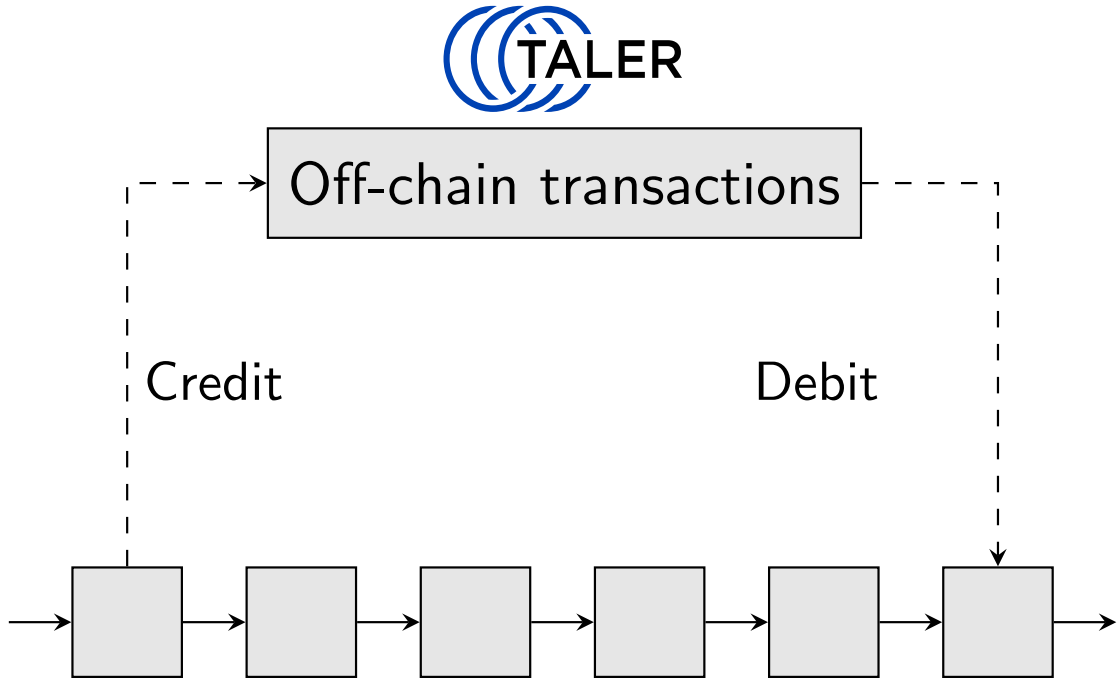
- Realtime transactions, 1 RTT
- Scalable microtransactions
- Blind signatures (privacy)

TALER

Taler

Blockchain settlement layer





Challenges

Taler Metadata

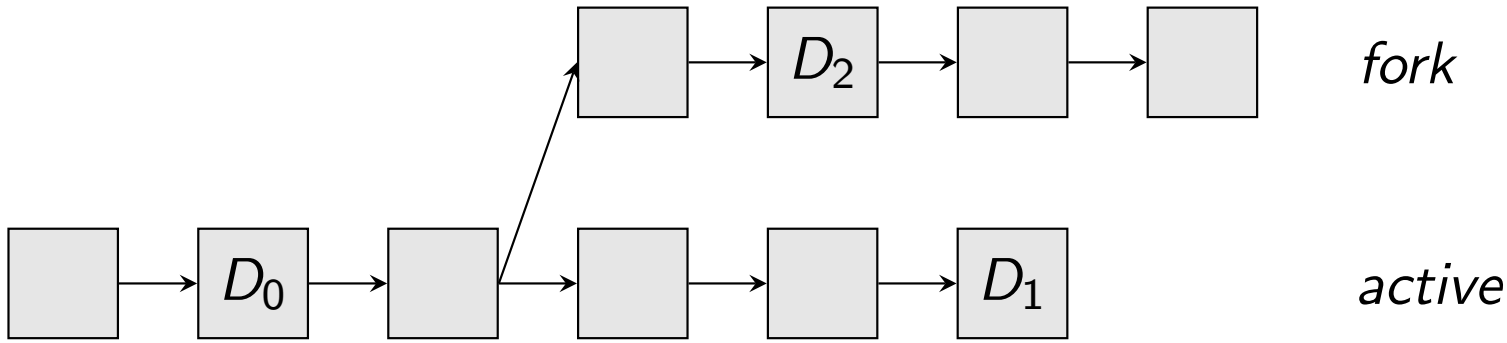
- Metadata are required to link a wallet to credits and allow merchant to link deposits to debits
- Putting metadata in blockchain transactions can be tricky

Blockchain based cryptocurrencies

- Blockchain transactions lack finality (fork)
- Transactions can be stuck for a long time (mempool)

Blockchain challenges

Chain reorganization

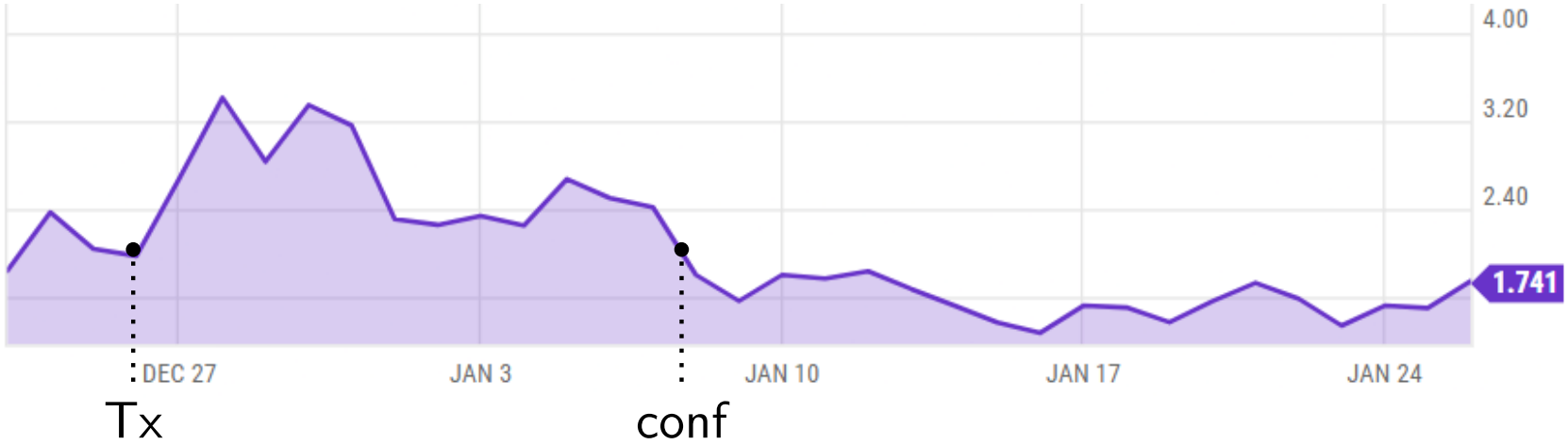


A fork is when concurrent blockchain states coexist. Nodes will follow the longest chain, replacing recent blocks if necessary during a blockchain reorganization. If a deposit transaction disappears from the blockchain, an irrevocable withdraw transactions would no longer be backed by credit.

Blockchain challenges

Stuck transactions

We want confirmed debits within a limited time frame.



When we trigger a debit with a fee too small, it may not be confirmed in a timely fashion.

Blockchain challenges

Stuck transactions

We want confirmed debits within a limited time frame.

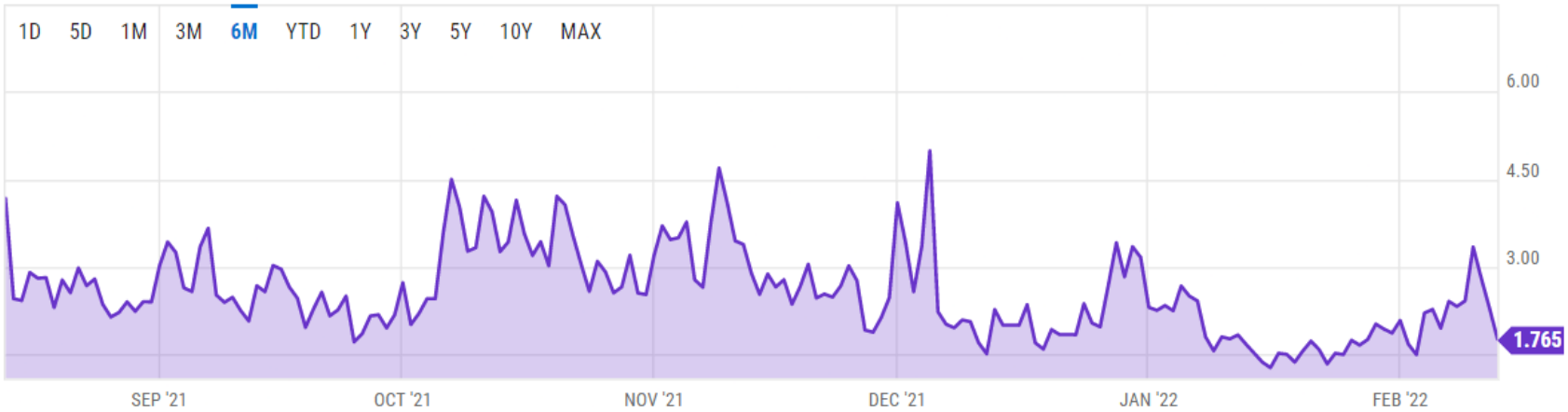
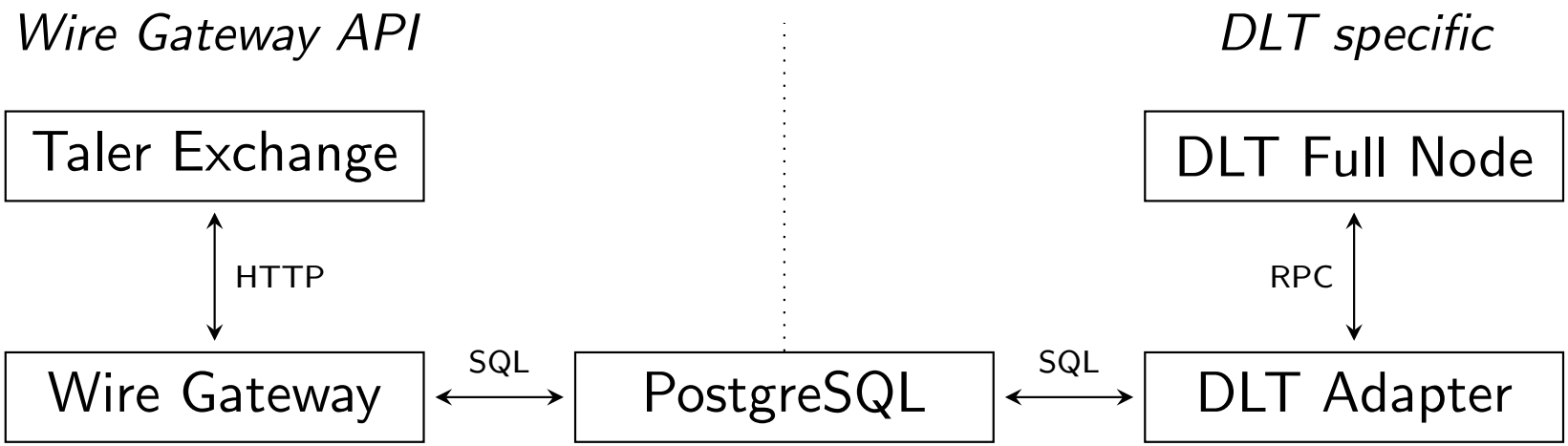


Figure: Bitcoin average transaction fee over 6 months (ychart)

However, transaction fees are unpredictable.

Depolymerization

Architecture



- Common database to store transactions state and communicate with notifications
- Wire Gateway for Taler API compatibility
- DLT specific adapter

Storing metadata

Bitcoin

Bitcoin - Credit

- Transactions from code
- Only 32B + URI
- **OP_RETURN**

Bitcoin - Debit

- Transactions from common wallet software
- Only 32B
- **Fake Segwit Addresses**

Storing metadata

Ethereum

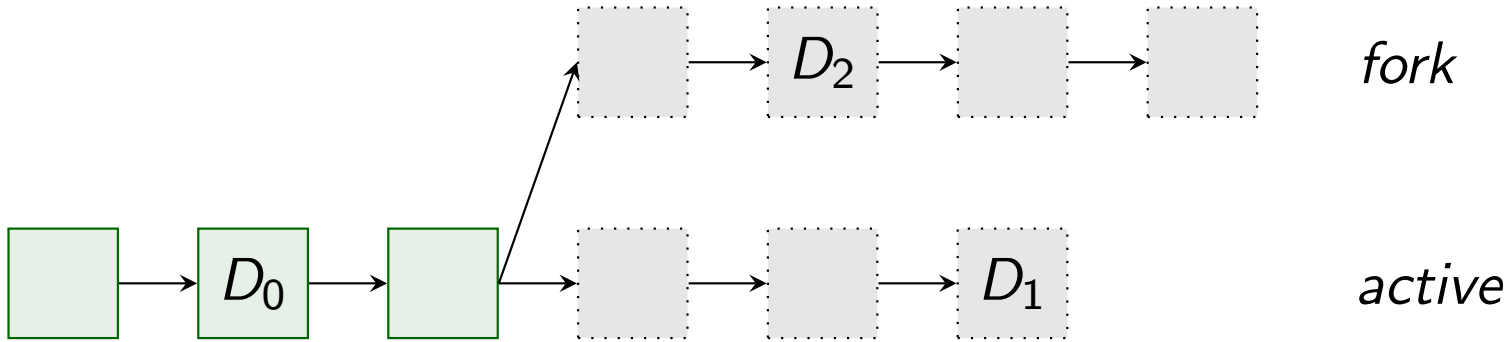
Smart contract ?

- Logs in smart contract is the recommend way (ethereum.org)
- Expensive (additional storage and execution fees)
- Avoidable attack surface (error prone)

Custom input format

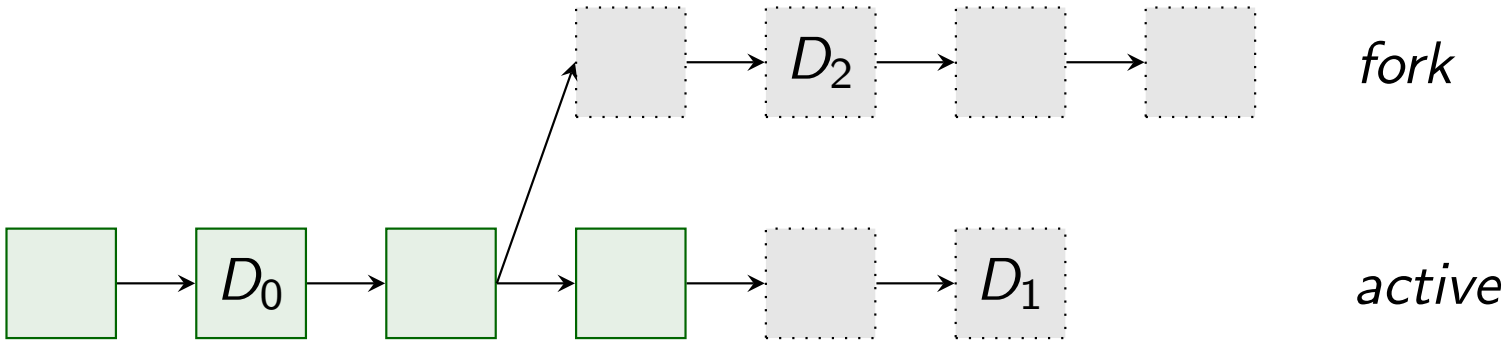
Use input data in transactions, usually used to call smart contract, to store our metadata.

Handling blockchain reorganization



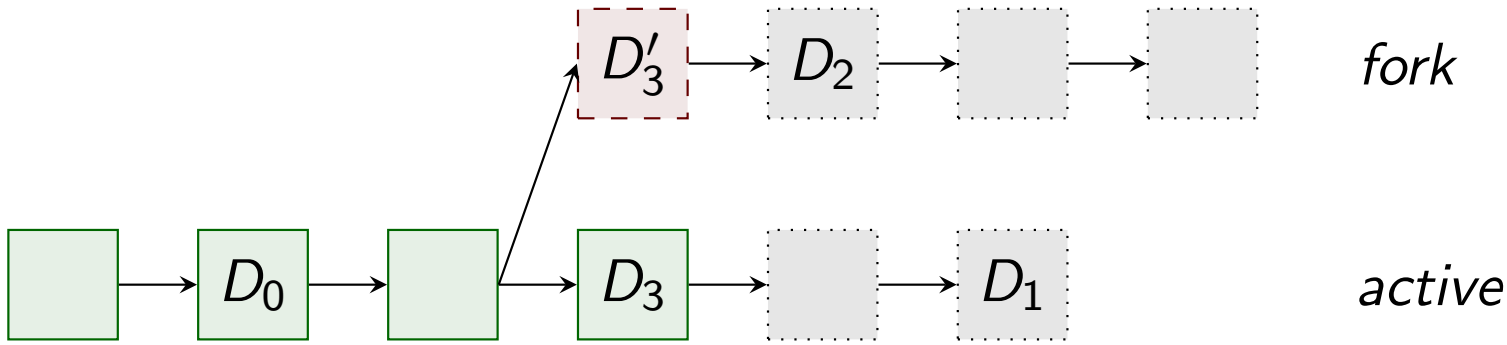
As small reorganizations are common, Satoshi already recommended to apply a confirmation delay to handle most disturbances and attacks.

Handling blockchain reorganization



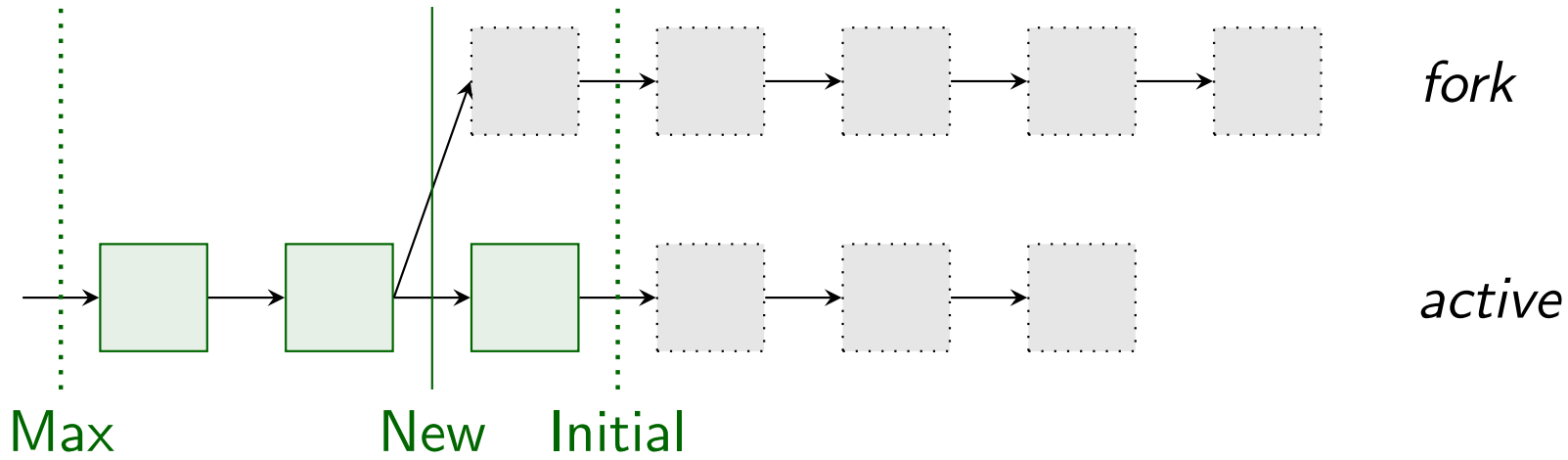
If a reorganization longer than the confirmation delay happens, but it did not remove credits, Depolymerizer is safe and automatically resumes.

Handling blockchain reorganization



If a fork removed a confirmed debit, an attacker may create a conflicting transaction. Depolymerizer suspends operation until lost credits reappear.

Adaptive confirmation



If we experience a reorganization once, its dangerously likely for another one of a similar scope to happen again. Depolymerizer learns from reorganizations by increasing its confirmation delay.

DLT Adapter

Architecture

Event system

- **Watcher** watch and notify for new blocks with credits
- **Wire Gateway** notify requested debits
- **Worker** operates on notifications updating state

DLT Adapter state machine

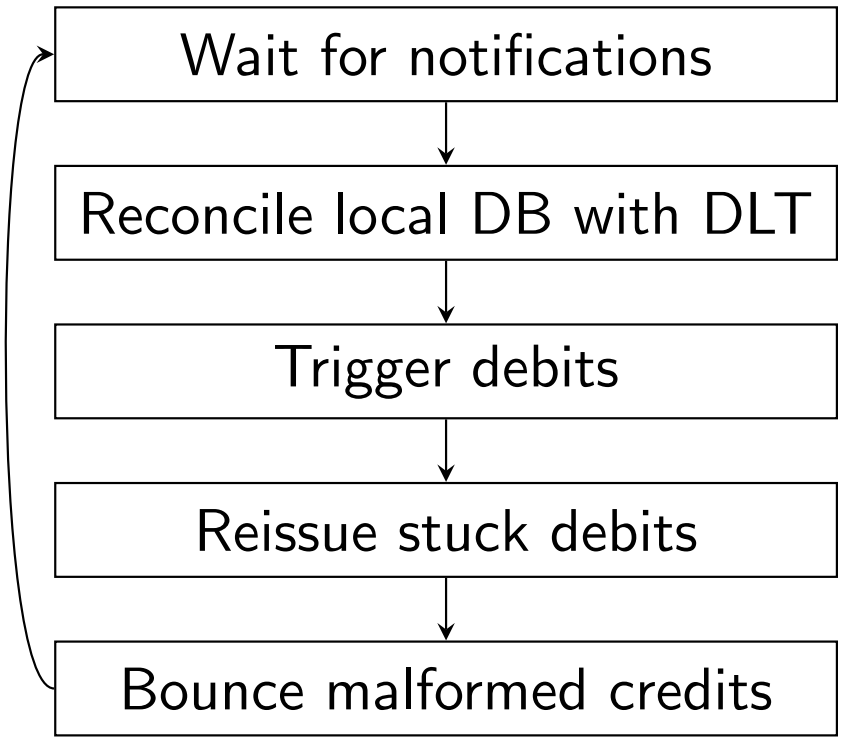


Figure: Worker loop

DLT reconcialisation

- List new and removed transactions since last reconciliation
- Check for confirmed credits removal
- Register new credits
- Recover lost debits

Related work

Centralization - Coinbase off-chain sending

+ Fast and cheap: off chain transaction
– Trust in Coinbase: privacy, security & transparency

Layering - Lightning Network

+ Fast and cheap: off-chain transactions
– Requires setting up bidirectional payment channels
– Fraud attempts are mitigated via a complex penalty system

Conclusion

Blockchains can be used as a settlement layer for GNU Taler with Depolymerizer.

- − Trust exchange operator or auditors
- \+ Fast and cheap
- \+ Realtime, ms latency
- \+ Linear scalability
- \+ Ecological
- \+ Privacy when it can, transparency when it must (avoid tax evasion and money laundering)

Future work:

- Universal auditability, using sharded transactions history
- Smarter analysis, update confirmation delay based on currency network behavior
- Multisig by multiple operator for transactions validation

TALER

Future Work & Conclusion

TALER

Taler: Project Status

https://docs.taler.net/

- Cryptographic protocols and core exchange component are stable
- Pilot project at Bern University of Applied Sciences cafeteria
- Regional currency projects in Switzerland preparing for launch
- Internal alpha deployment with GLS Bank (Germany)

TALER

Competitor comparison

	Cash	Bitcoin	Zerocoin	Creditcard	GNU Taler
Online	−−−	++	++	+	+++
Offline	+++	−−	−−	+	++
Trans. cost	+	−−−	−−−	−	++
Speed	+	−−−	−−−	o	++
Taxation	−	−−	−−−	+++	+++
Payer-anon	++	o	++	−−−	+++
Payee-anon	++	o	++	−−−	−−−
Security	−	o	o	−−	++
Conversion	+++	−−−	−−−	+++	+++
Libre	−	+++	+++	− − −	+++

TALER

Active collaborations

Freie Universität Berlin:
Programmability & embedded systems

The GNU Project:
Integration into FLOSS software

Fraunhofer Gesellschaft:
Identity management & SSI & wallet-to-wallet communication

NGI TALER:
11 partners deploying GNU Taler across Europe

NGI TALER PILOT

https://taler.net/en/consortium.html

- EU Project started December 2023 to deploy GNU Taler
- 3 financial institutions (GLS Bank, Magnet Bank, Visual Vest), 2 academic institutions (Berner FH, TU Eindhoven), 3 SMEs (Taler Systems SA, Code Blau GmbH, Petit Singularites), 3 non-profits (NLnet Foundation, E-Seniors Association, Homo Digitalis)
- $\approx$ €5M budget over 3 years
- Objective: **Deploy GNU Taler in Europe**

TALER

Key NGI PILOT Activities

- Integration (core banking, online publishers, e-commerce, public transportation)
- Compliant (establish compliance processes at each bank)
- Availability (packaging, porting to more platforms, browsers)
- Hardware support (offline payments, vending machines)
- Security audits of code and design
- Accessible (old people, children, blind users)
- Future-proof (post-quantum, standardized)
- Widely known and used (community building, open calls)

Launch Timeline

Q2'2022 Internal deployment at BFH

Q3'2024 Deployment of local currency Netzbon in Basel

Q4'2024 Public deployment of eCHF stablecoin in Switzerland, cleared by FINMA

Q1'2025 GLS bank launches in Eurozone

Q3'2025 Magnet bank launches in Hungary

Other ongoing developments

- Privacy-preserving auctions (trading, currency exchange) (`oezguer@taler.net`)
- Hardware and software support for embedded systems (`mikolai@taler.net`)
- GNU Name System registry with GNU Taler payments (`schanzen@gnunet.org`)
- Performance improvements for RSA in FLOSS crypto libraries (NLnet project)
- Parallel verification of RSA signatures on GPUs (libgpuverify.git)
- Tax-deductable receipts for donations to charities (donau.git)
- Unlinkable anonymous subscriptions and discount tokens (merchant.git, branch)
- Support for illiterate and innumerate users[7] (not yet funded)

[7] Background: `https://myoralvillage.org/`

How to support?

Join: https://lists.gnu.org/mailman/listinfo/taler

Discuss: https://ich.taler.net/

Develop: https://bugs.taler.net/,
https://git.taler.net/

Apply: https://nlnet.nl/propose, https://nlnet.nl/taler

Translate: https://weblate.taler.net/,
translation-volunteer@taler.net

Integrate: https://docs.taler.net/

Donate: https://gnunet.org/ev

Partner: https://taler-systems.com/

Conclusion

What can we do?

- Suffer mass-surveillance enabled by credit card oligopolies with high fees, and
- Engage in arms race with deliberately unregulatable blockchains

OR

- Establish free software alternative balancing social goals!

TALER

Do you have any questions?

References:

1. Özgür Kesim, Christian Grothoff, Florian Dold and Martin Schanzenbach. *Zero-Knowledge Age Restriction for GNU Taler*. **27th European Symposium on Research in Computer Security (ESORICS), 2022**.
2. David Chaum, Christian Grothoff and Thomas Moser. *How to issue a central bank digital currency*. **SNB Working Papers, 2021**.
3. Christian Grothoff, Bart Polot and Carlo von Loesch. *The Internet is broken: Idealistic Ideas for Building a GNU Network*. **W3C/IAB Workshop on Strengthening the Internet Against Pervasive Monitoring (STRINT)**, 2014.
4. Jeffrey Burdges, Florian Dold, Christian Grothoff and Marcello Stanisci. *Enabling Secure Web Payments with GNU Taler*. **SPACE 2016**.
5. Florian Dold, Sree Harsha Totakura, Benedikt Müller, Jeffrey Burdges and Christian Grothoff. *Taler: Taxable Anonymous Libre Electronic Reserves*. Available upon request. 2016.
6. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer and Madars Virza. *Zerocash: Decentralized Anonymous Payments from Bitcoin*. **IEEE Symposium on Security & Privacy, 2016**.
7. David Chaum, Amos Fiat and Moni Naor. *Untraceable electronic cash*. **Proceedings on Advances in Cryptology, 1990**.
8. Phillip Rogaway. *The Moral Character of Cryptographic Work*. **Asiacrypt**, 2015.

Let money facilitate trade; but ensure capital serves society.

TALER

Rights

- GNUnet e.V. shared copyrights of their AGPLv3+ licensed code with Taler Systems SA
- Taler Systems SA holds copyrights to entire GNU Taler code base (AGPLv3+, GPLv3+, dual-licensing exclusive domain of Taler Systems SA)
- Taler Systems SA applied for patent on offline payment solution
- Taler Systems SA holds trademark on “Taler”.
- FSF holds trademark on “GNU”, we are authorized to use “GNU Taler”.
- Taler Systems SA owns `taler.net` and `taler-systems.com`.

TALER