GNU

taler.net

taler@twitter

taler-systems.com

Florian Dold &
Christian Grothoff

{dold,grothoff}@taler.net

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?"*

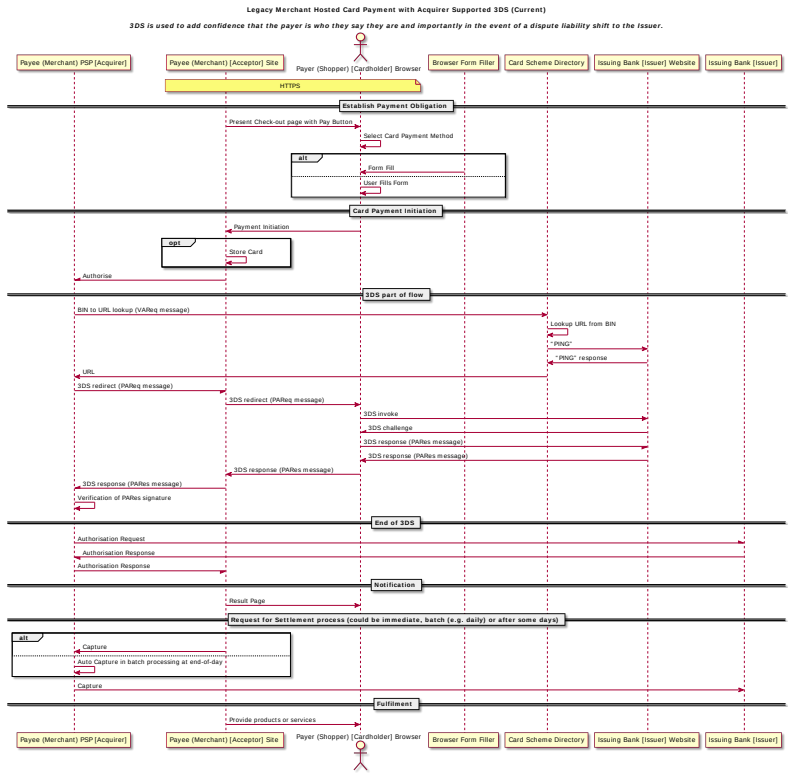
Mastercard/Visa are too transparent.

"I think one of the big things that we need to do, is we need to get a way 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)

The Bank's Problem

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

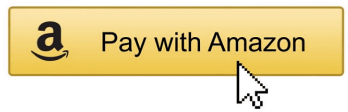
- Complicated process
- Shifts liability to consumer
- Significant latency
- Can refuse valid requests
- Legal vendors excluded
- No privacy for buyers



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 sovereingity is in danger



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:

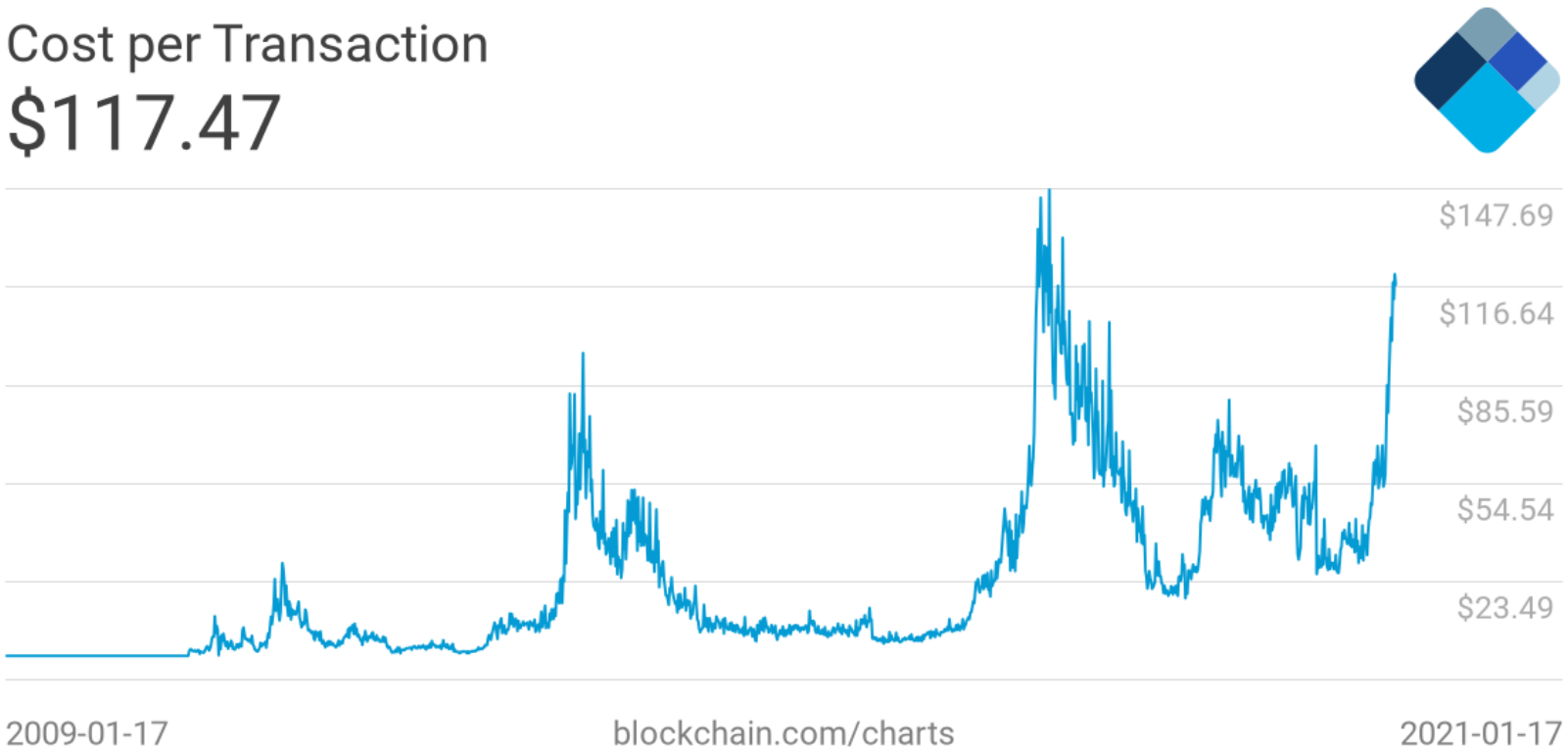
  $\Rightarrow$ lack of regulation is a feature!
- Implemented in free software
- Decentralised peer-to-peer system
- Decentralised banking requires solving Byzantine consensus
- Creative solution: tie initial accumulation to solving consensus

  $\Rightarrow$ Proof-of-work advances ledger

  $\Rightarrow$ Very expensive banking

?

Background: `https://blockchain.info/`



Current average transaction value: $\approx$ 1000 USD

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?

Digital cash, made **socially responsible**.

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

What is Taler?

Taler is an electronic instant payment system.

- Uses electronic coins stored in **wallets** on customer's device
- Like **cash**
- Pay in **existing currencies** (i.e. EUR, USD, BTC), or use it to create new **regional currencies**

What is Taler?

Taler is an electronic instant payment system.

- Uses electronic coins stored in **wallets** on customer's device
- Like **cash**
- Pay in **existing currencies** (i.e. EUR, USD, BTC), or use it to create new **regional currencies**

However, Taler is

- *not* a currency
- *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
- *not* a speculative asset / "get-rich-quick scheme"

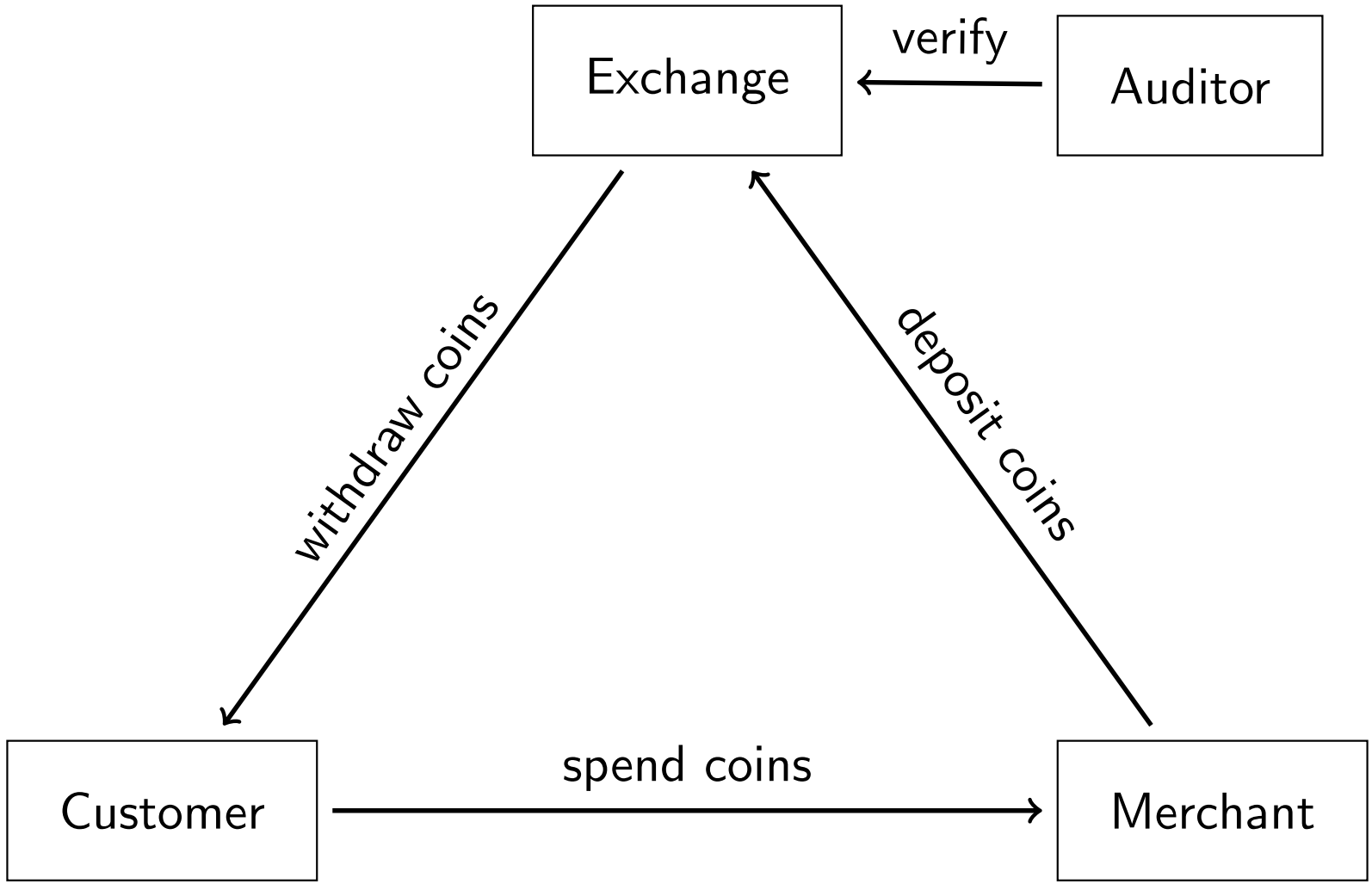
Design principles

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

GNU Taler must ...

1. ... be implemented as **free software**.
2. ... protect the **privacy of buyers**.
3. ... must 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 Overview



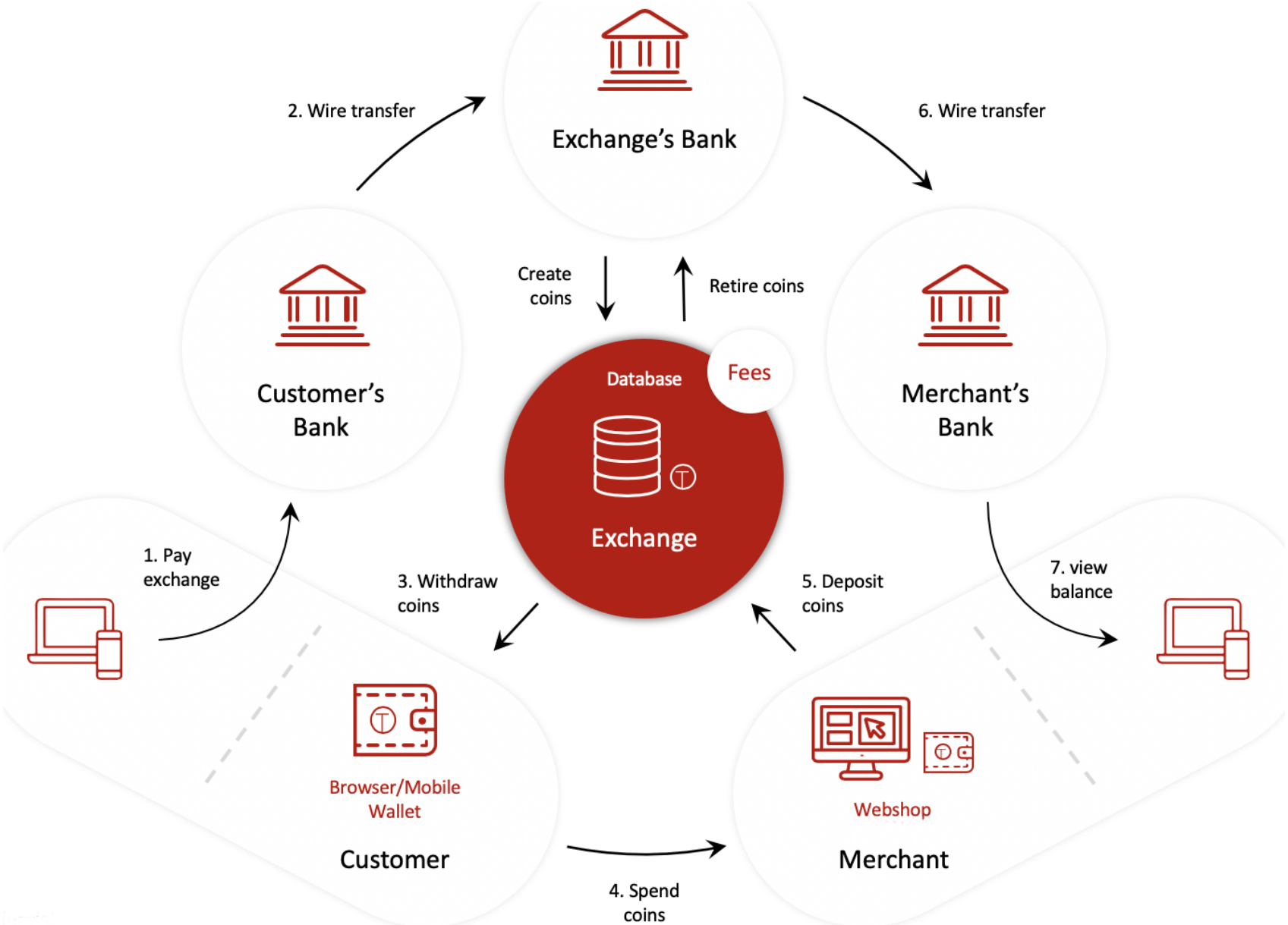
The Taler Software Ecosystem

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

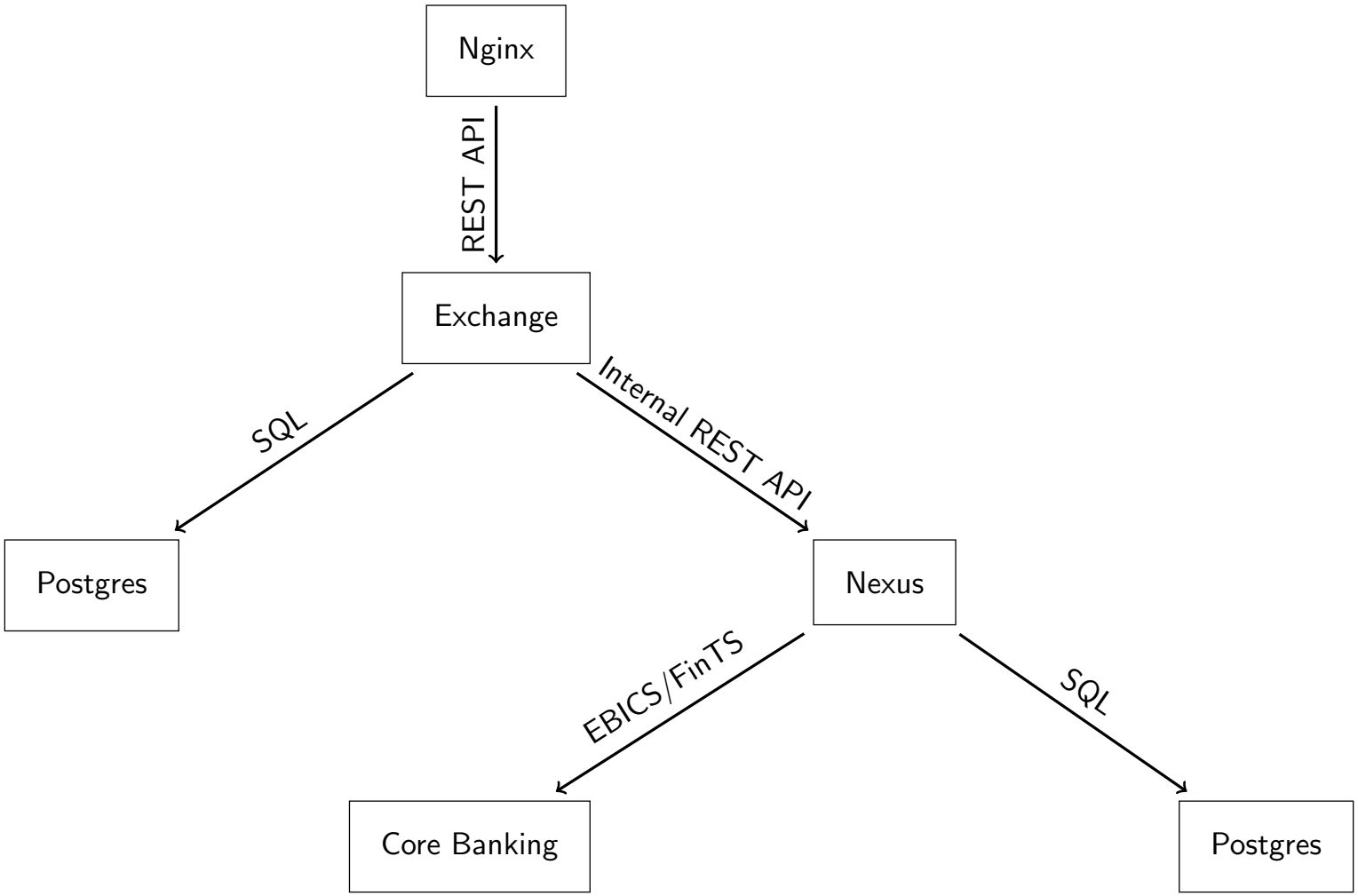
Taler is based on modular components that work together to provide a complete payment system:

- **Exchange:** Service provider for digital cash
  - Core exchange software (cryptography, database)
  - Air-gapped key management, real-time **auditing**
  - LibEuFin: Modular integration with banking systems
- **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
  - **Anastasis**: Recovery of lost wallets based on secret splitting

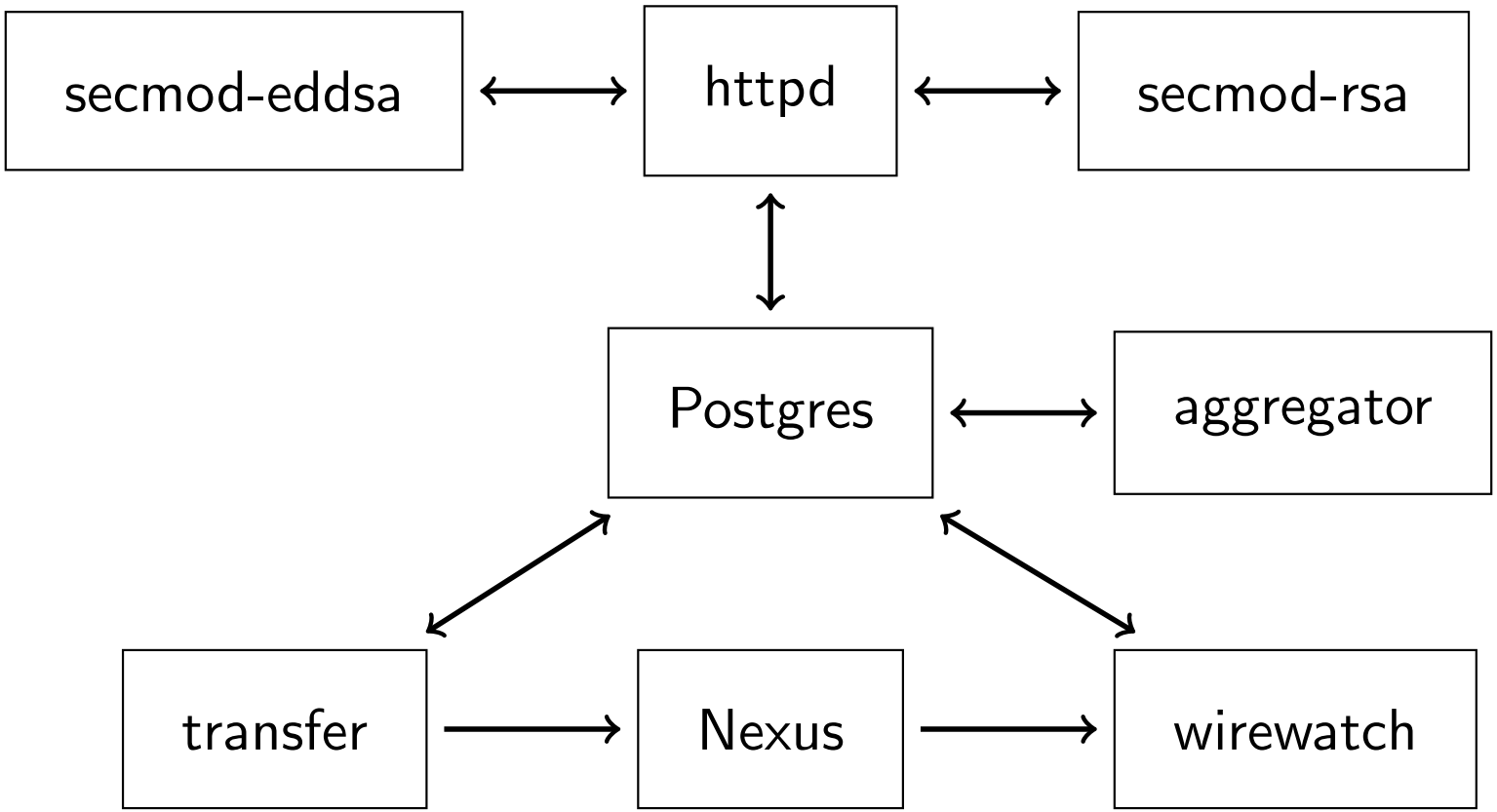
Architecture of Taler



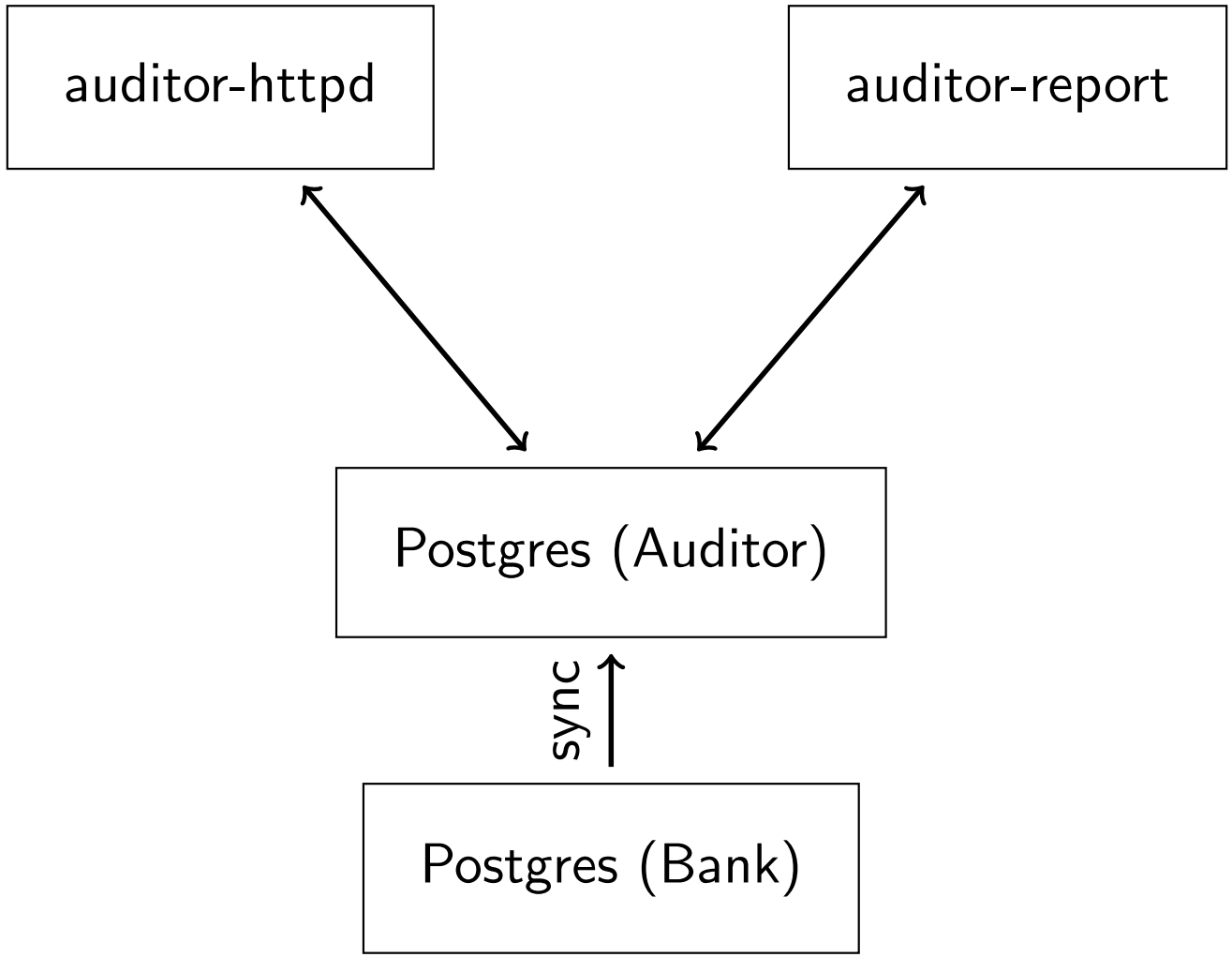
Taler: Bank Perspective



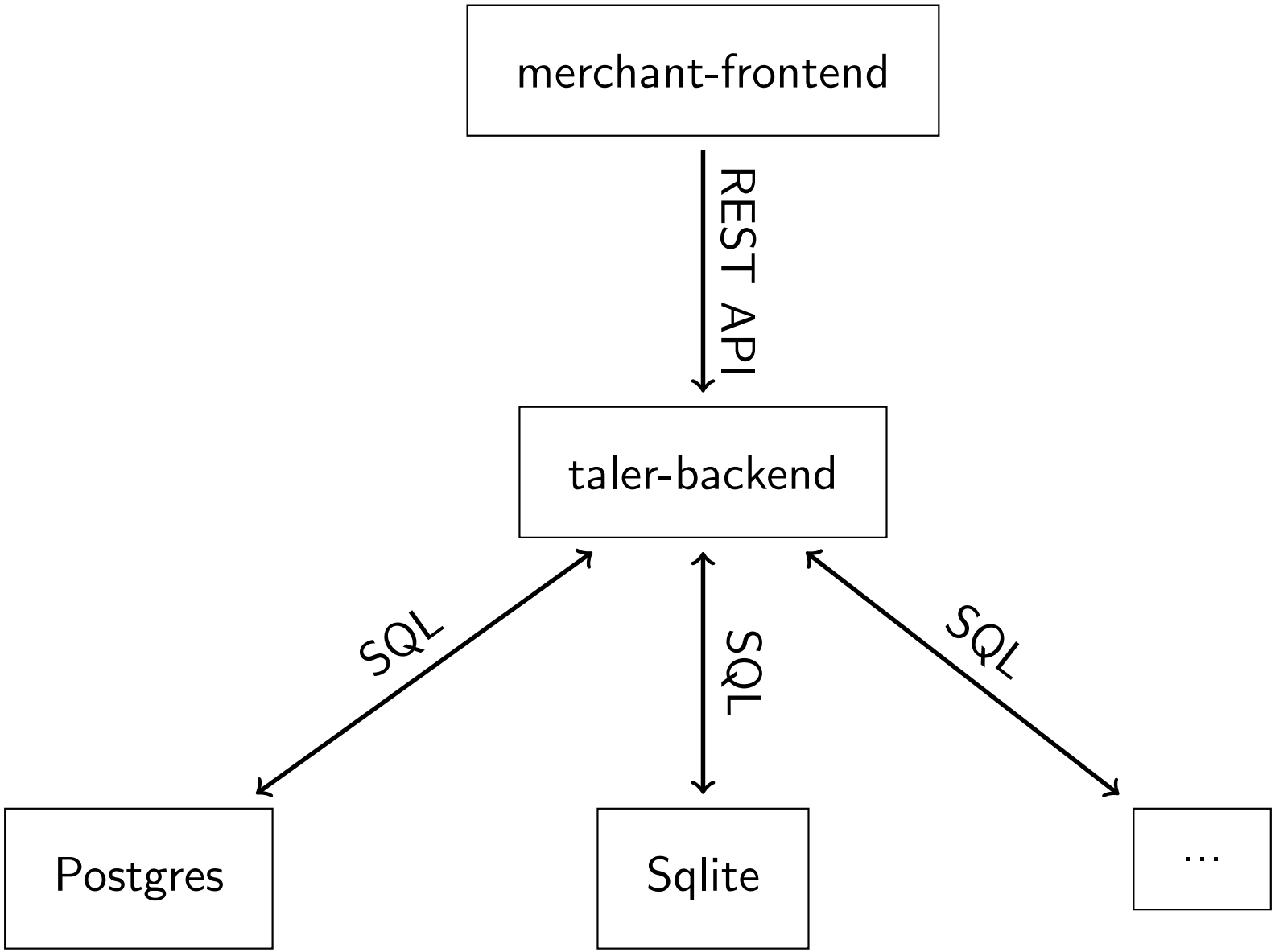
Taler: Exchange Architecture



Taler: Auditor Perspective

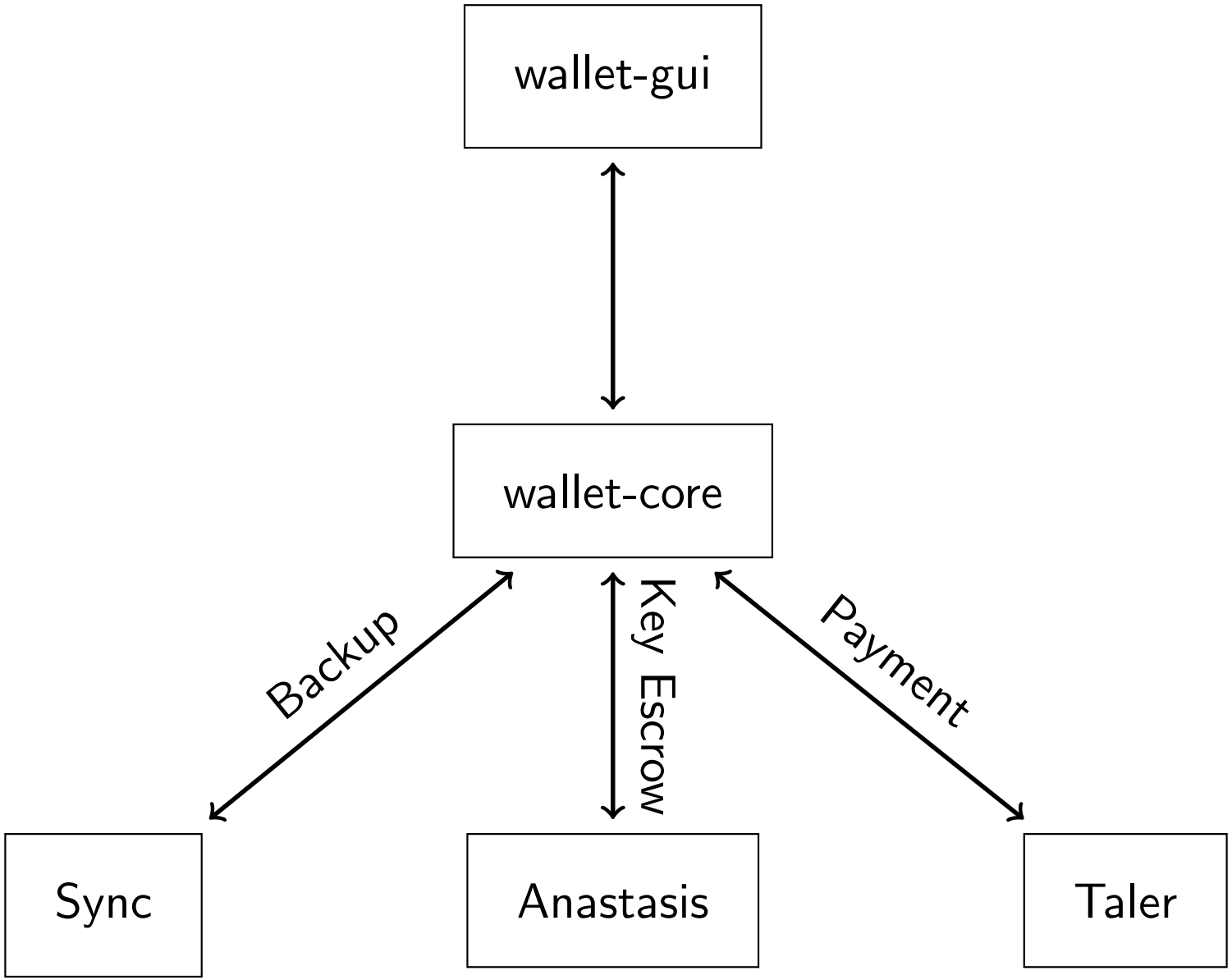


Taler: Merchant Perspective



Taler: Wallet Architecture

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



Taler: Unique Regulatory Features for Central Banks

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

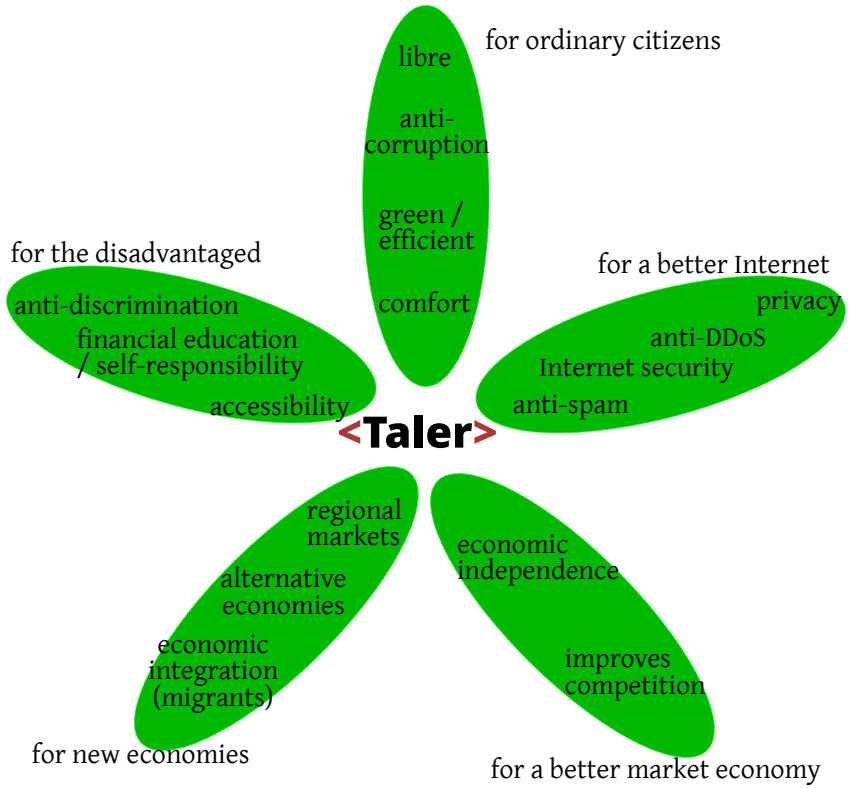
- Central bank issues digital coins equivalent to issuing cash
  ⇒ monetary policy remains under CB control
- Architecture with consumer accounts at commercial banks
  ⇒ no competition for commercial banking (S&L)
  ⇒ CB does not have to manage KYC, customer support
- Withdrawal limits and denomination 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

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.

Social Impact of 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

Use Cases: Refugee Camps

Today:

- Non-bankable
- Direct distribution of goods to population
- Limited economic activity in camps
- High level of economic dependence

Use Cases: Refugee Camps

Today:

- Non-bankable
- Direct distribution of goods to population
- Limited economic activity in camps
- High level of economic dependence

With GNU Taler:

- Local currency issued as basic income backed by aid
- Taxation possible based on economic status
- Local governance enabled by local taxes
- Increased economic independence and political participation

Use Case: Anti-Spam

Background: `https://pep.security/`

Today, p≡p provides authenticated encryption for e-mail:

- Free software
- Easy to use opportunistic encryption
- Available for Outlook, Android, Enigmail
- Spies & spam filters can no longer inspect content

Use Case: Anti-Spam

Background: `https://pep.security/`

Today, p≡p provides authenticated encryption for e-mail:

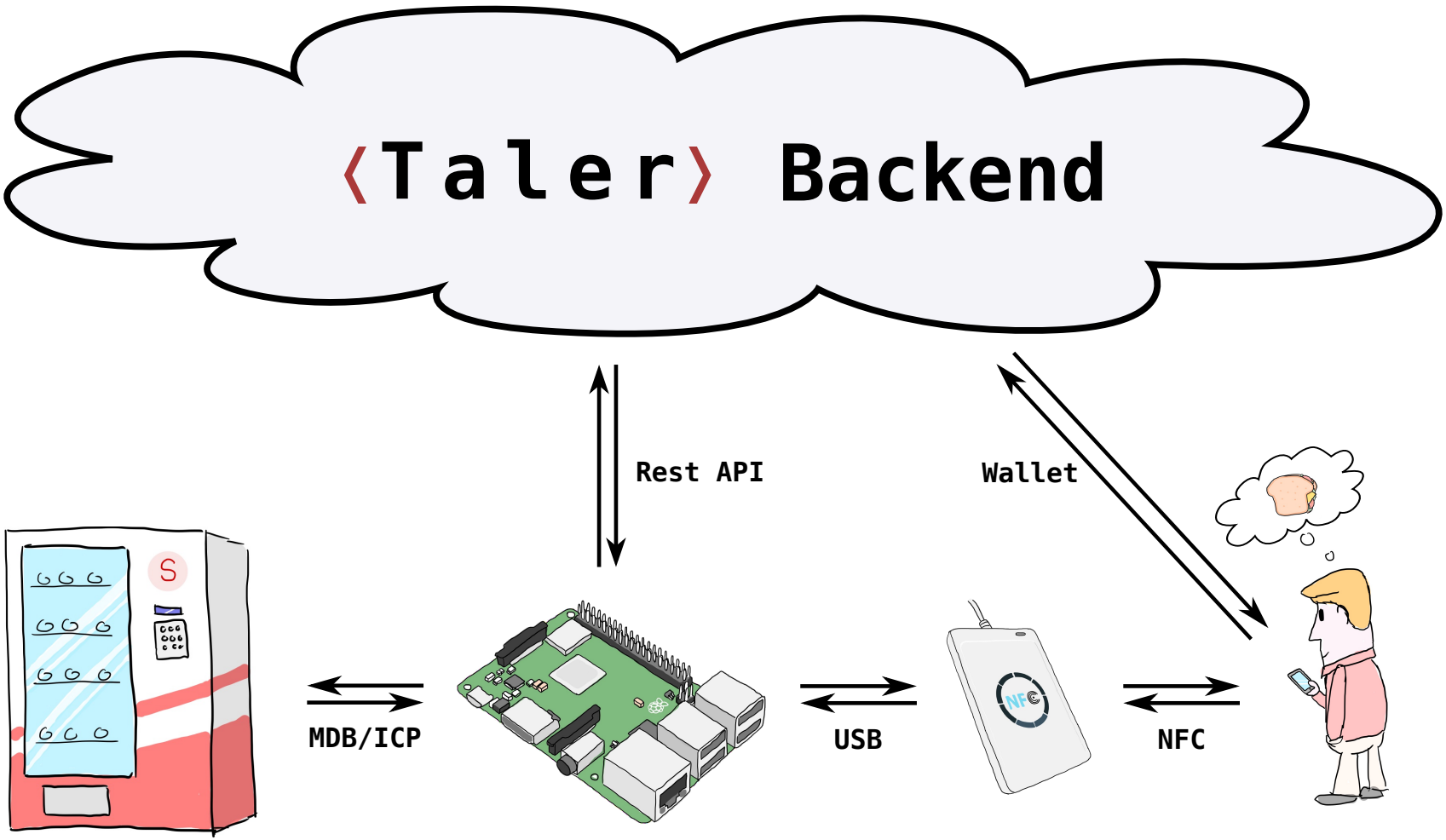
- Free software
- Easy to use opportunistic encryption
- Available for Outlook, Android, Enigmail
- Spies & spam filters can no longer inspect content

With GNU Taler:

- Peer-to-peer payments via e-mail
- If unsolicited sender, hide messages from user & automatically request payment from sender
- Sender can attach payment to be moved to inbox
- Receiver may grant refund to sender

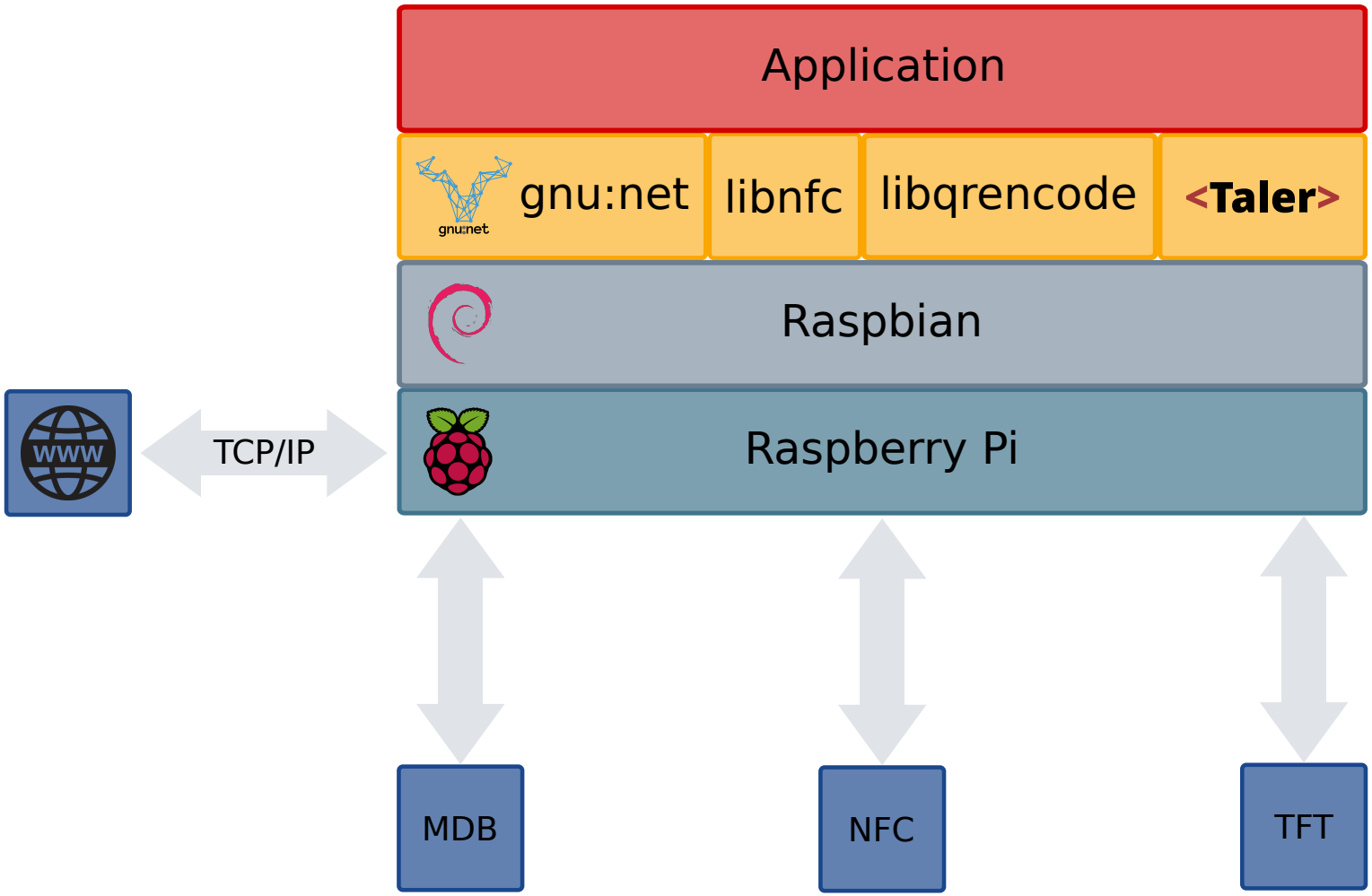
Example: The Taler Snack Machine[1]

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

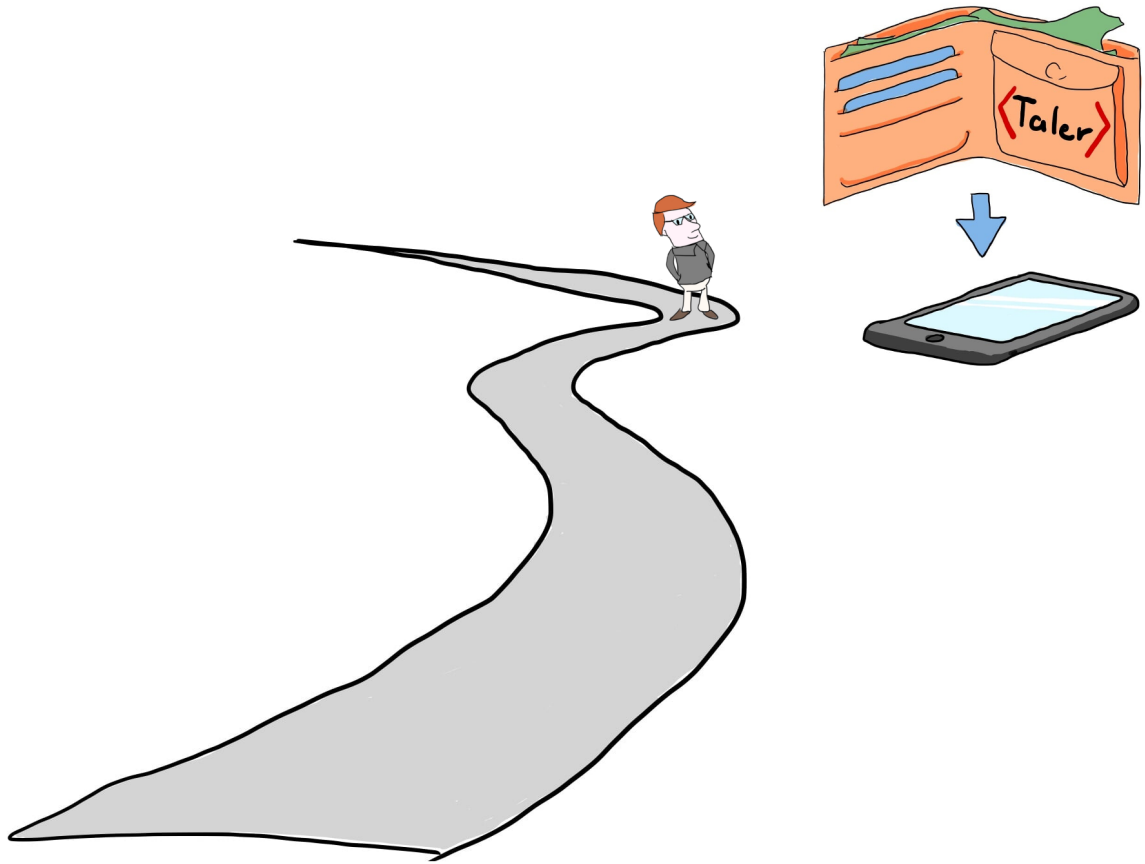


[1] By M. Boss and D. Hofer

Software architecture for the Taler Snack Machine

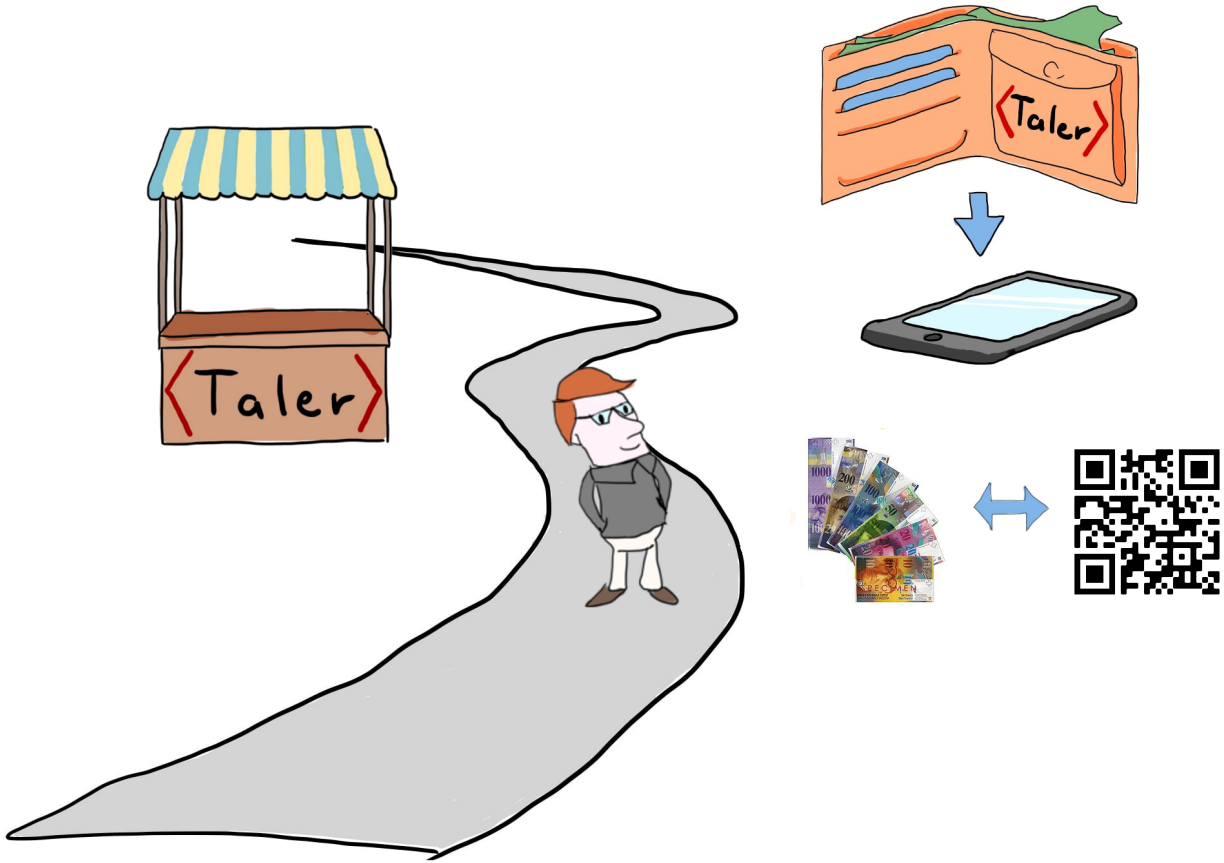


User story: Install App on Android[2]



[2] https://wallet.taler.net/

User story: Withdraw e-cash



User story: Use machine!

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:

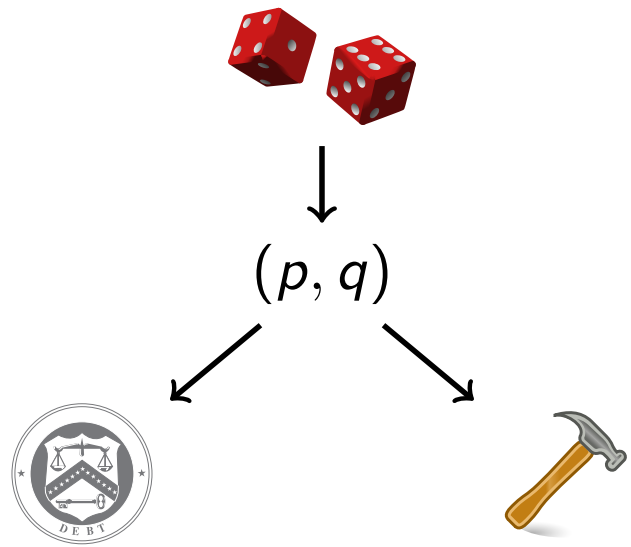
- Merchant's income is visible from deposits.
- Hash of contract is part of deposit data.
- State can trace income and enforce taxation.

Limitations:

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

Exchange setup: Create a denomination key (RSA)

1. Pick 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)

- pick random m mod o as private key
- $M = mG$ public key

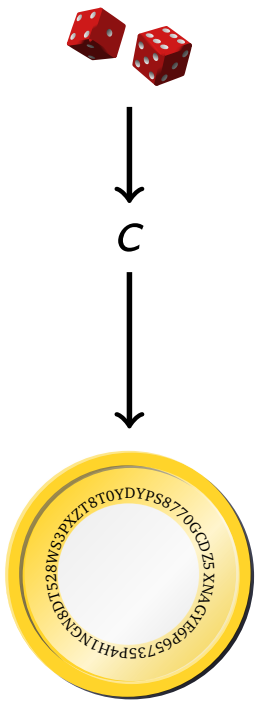
m

M

Capability: $m \Rightarrow$ 

Customer: Create a planchet (EdDSA)

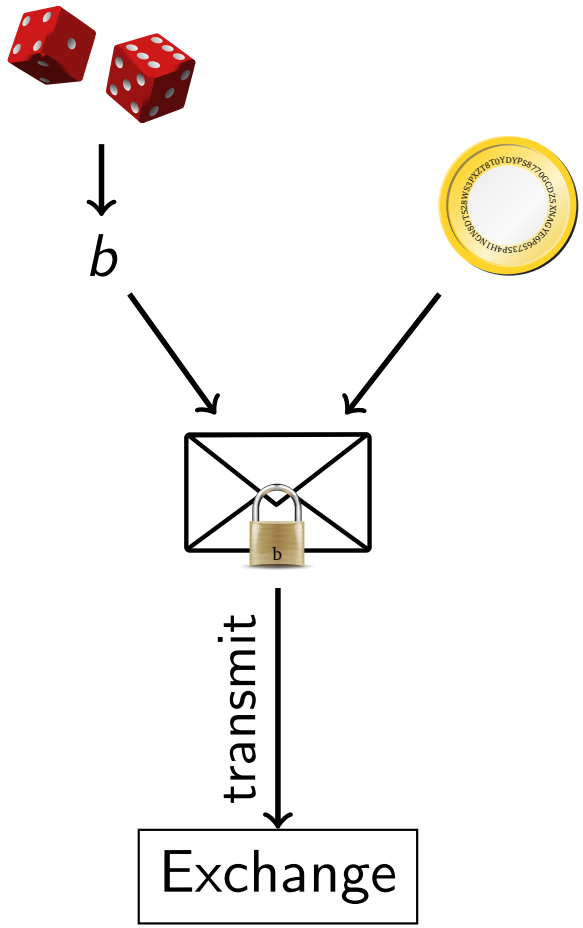
- Pick random c mod o private key
- $C = cG$ public key



Capability: $c \Rightarrow$ 

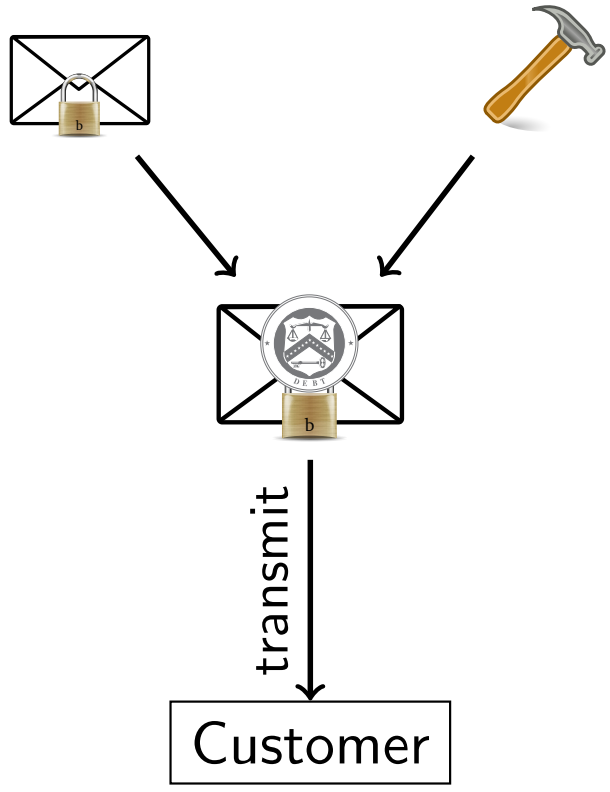
Customer: Blind planchet (RSA)

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



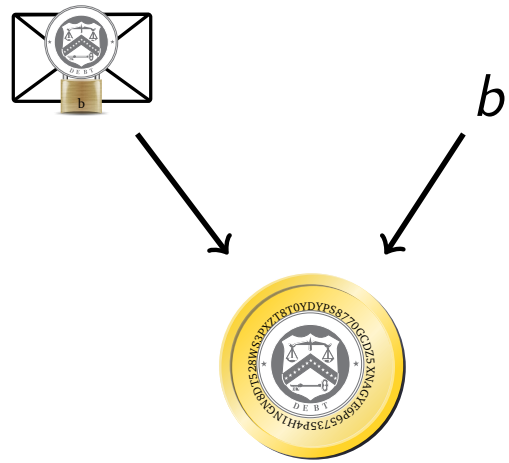
Exchange: Blind sign (RSA)

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

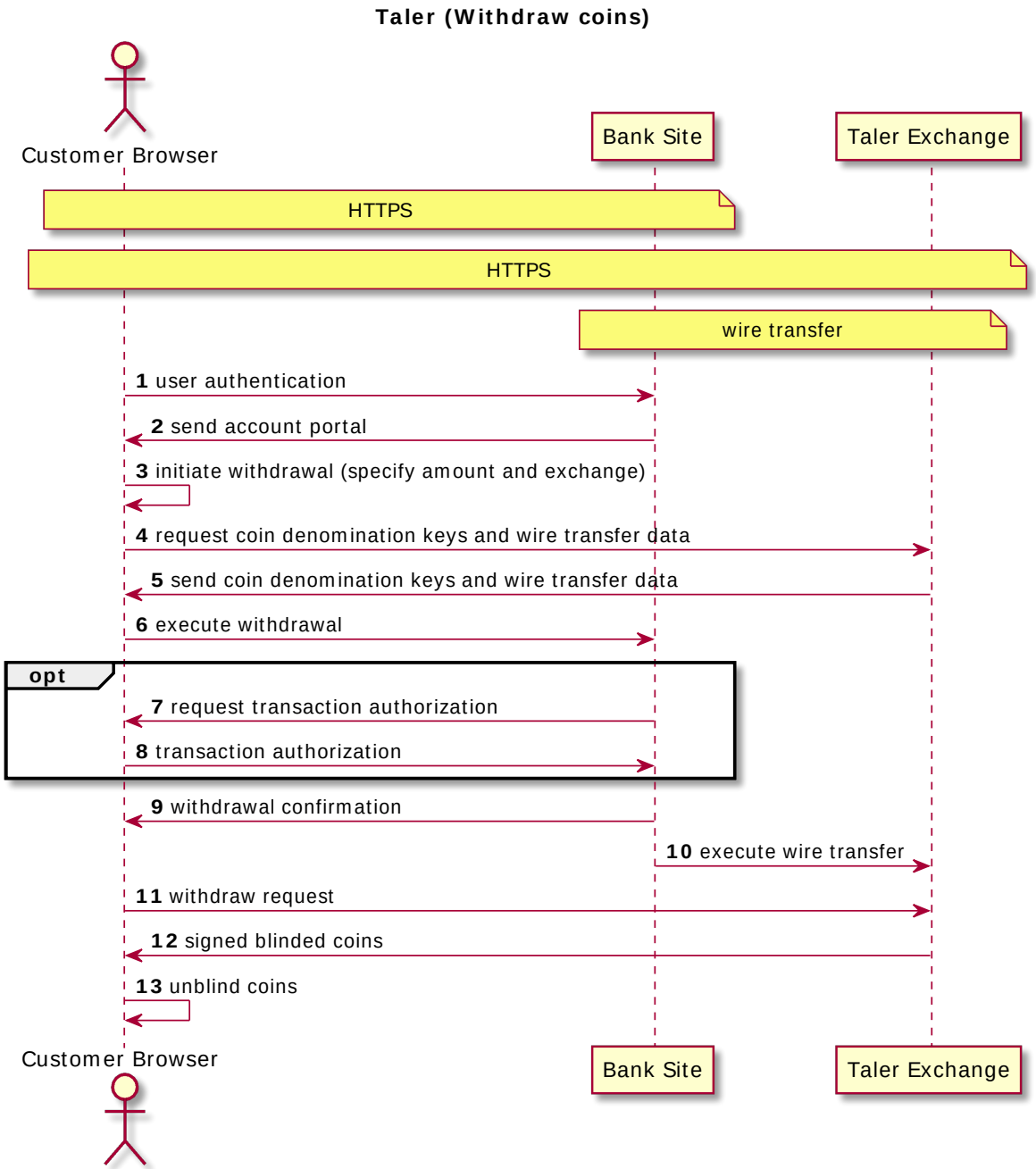


Customer: Unblind coin (RSA)

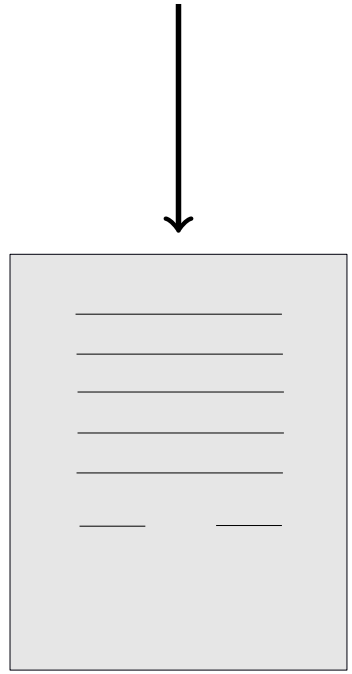
1. Receive s'.
2. Compute $s := s'b^{-1} \mod n$



Withdrawing coins on the Web



Customer: Build shopping cart



transmit

Merchant

Merchant Integration: Payment Request

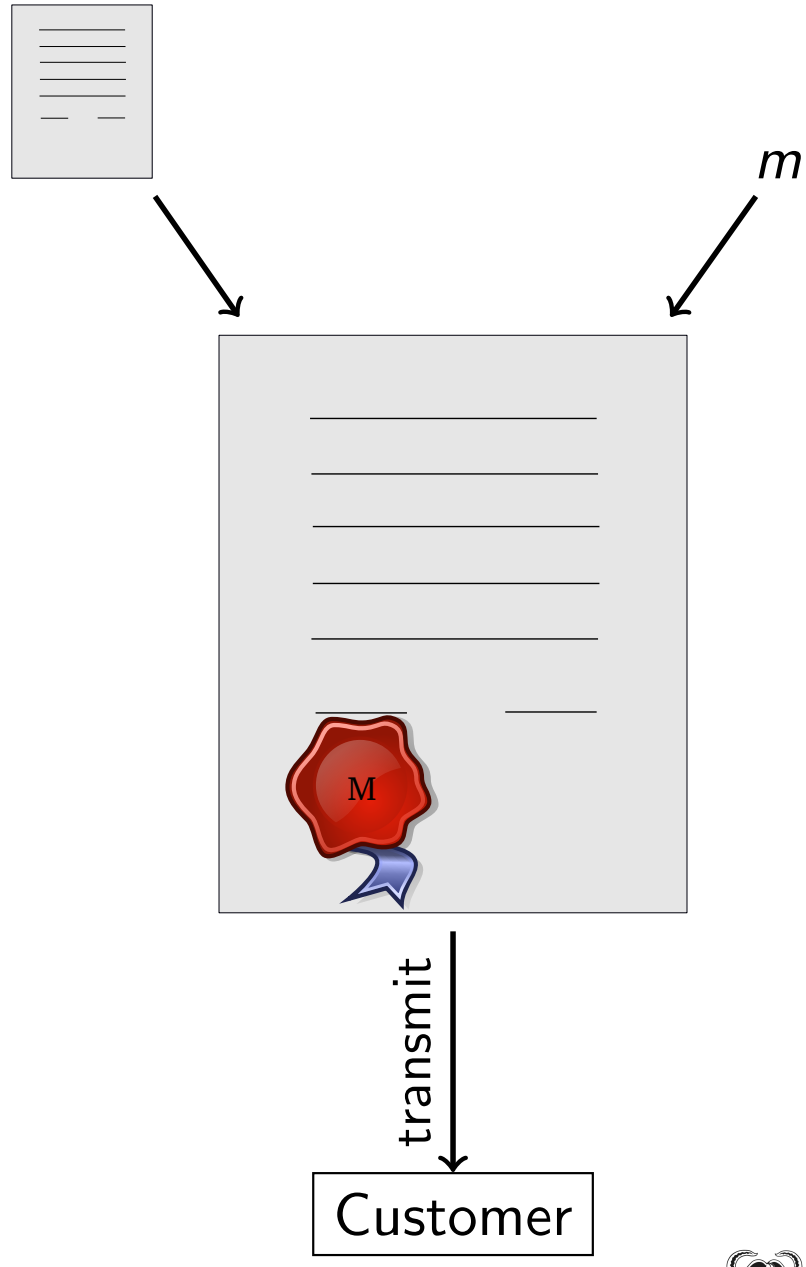
```
HTTP/1.1 402 Payment Required
Content-Type: text/html; charset=UTF-8
X-Taler-Contract-Url: https://shop/generate-contract/42

<!DOCTYPE html>
<html>
  <!-- fallback for browsers without the Taler extension -->
  You do not seem to have Taler installed, here are other
  payment options ...
</html>
```

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
}
```

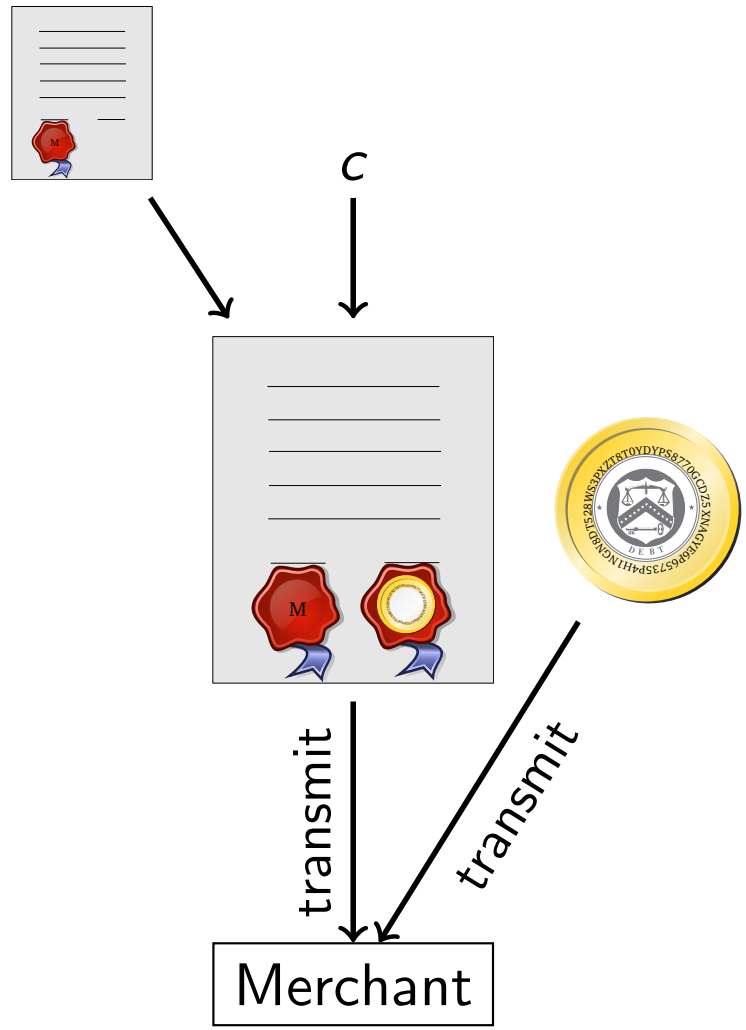
Merchant: Propose contract (EdDSA)

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



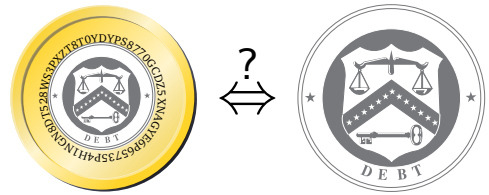
Customer: Spend coin (EdDSA)

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



Merchant and Exchange: Verify coin (RSA)

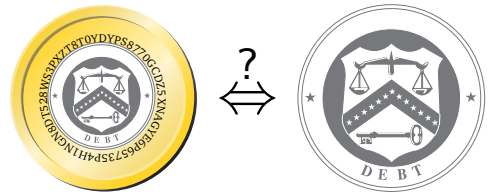
$$s^e \stackrel{?}{\equiv} FDH(C) \mod n$$



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.

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.

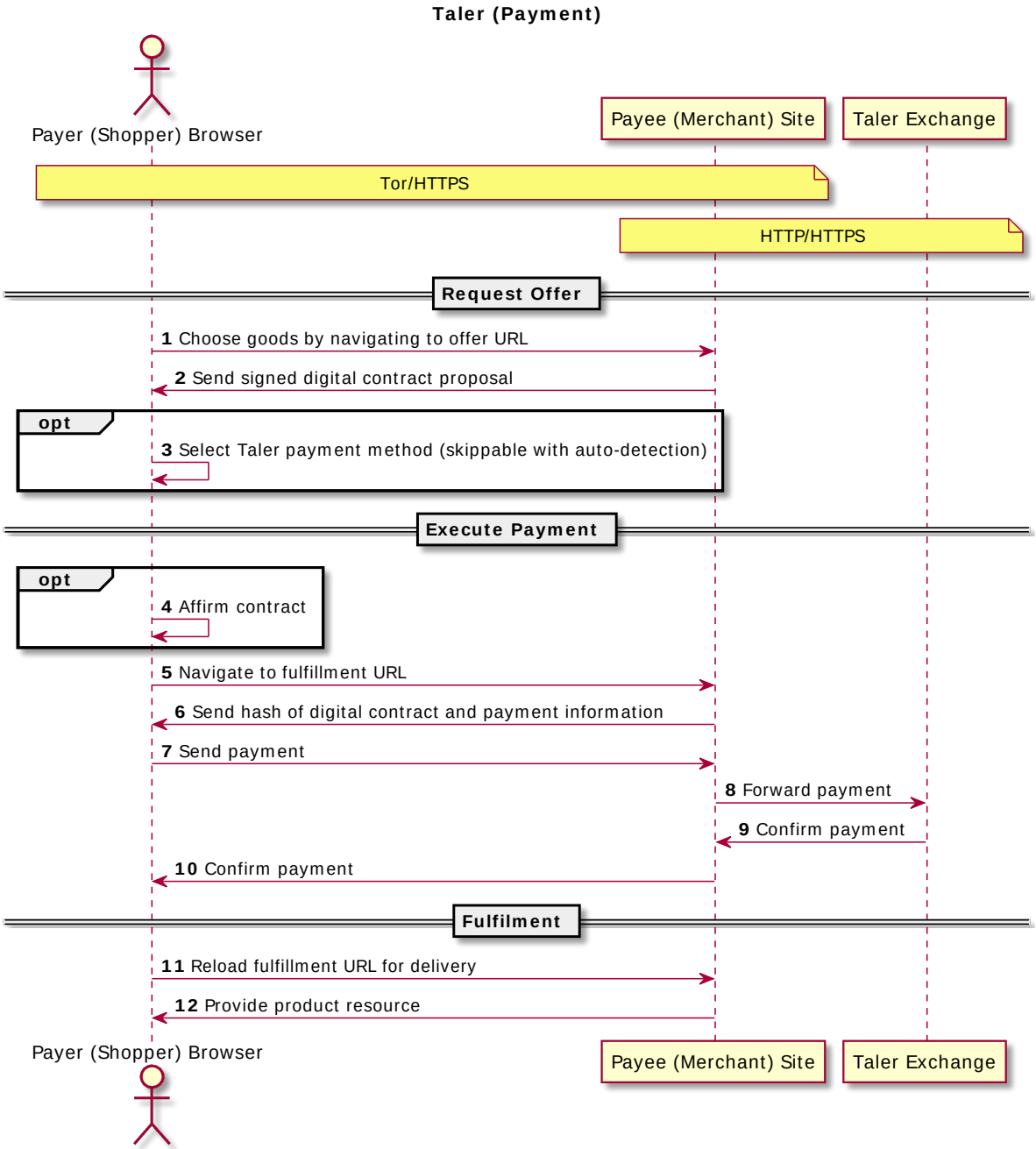
$\Rightarrow$ Permanent offline features weaken a digital payment solution (privacy, security)

$\Rightarrow$ 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)

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:

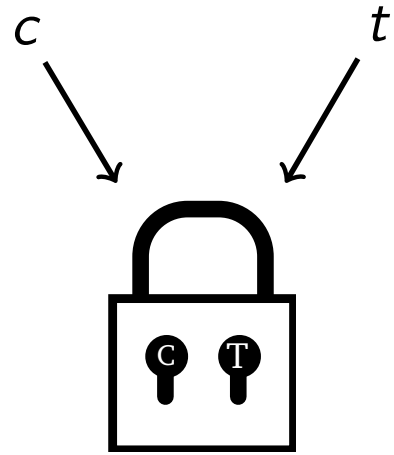
- maintain unlinkability
- maintain taxability of transactions

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. Define $C = cG$
3. Define $T = tG$
4. Compute DH
   $cT = c(tG) = t(cG) = tC$

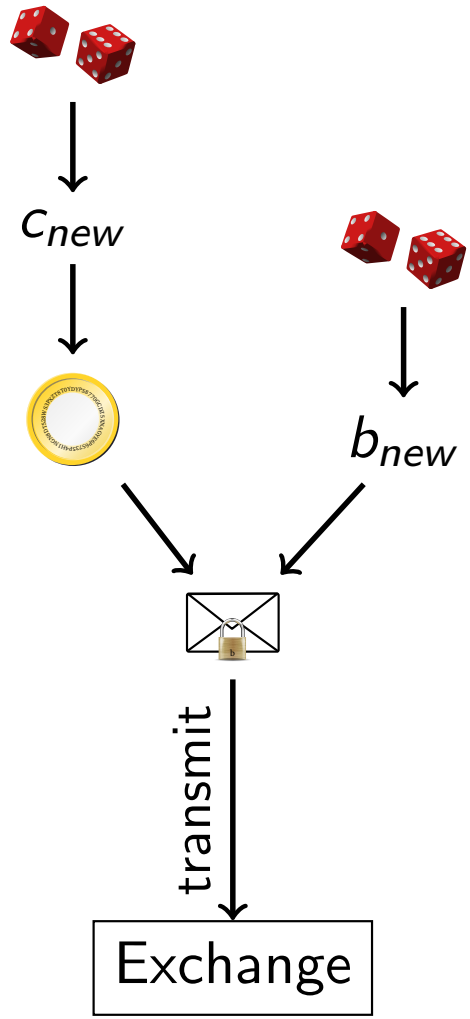


Strawman solution

Given partially spent private coin key c_{old}:

1. Pick random $c_{new} \mod o$ private key
2. $C_{new} = c_{new} G$ public key
3. Pick 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}.

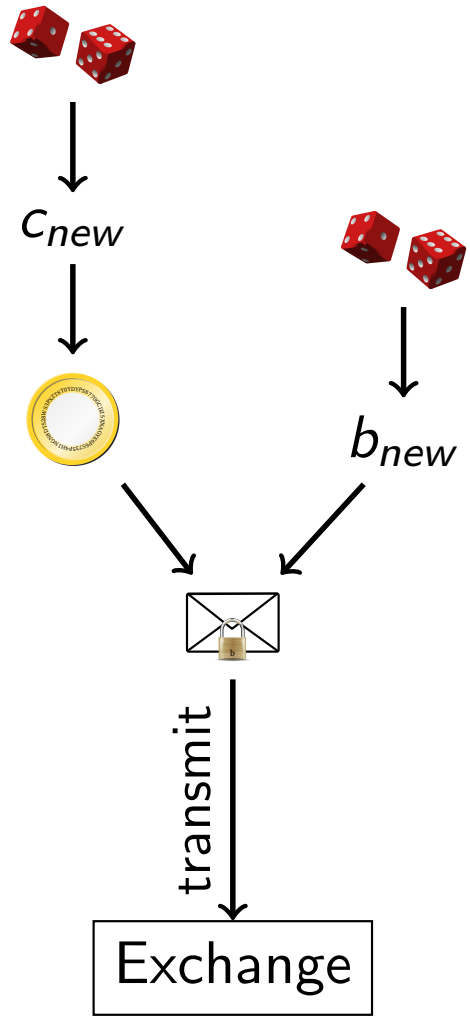


Strawman solution

Given partially spent private coin key c_{old}:

1. Pick random c_{new} mod o private key
2. $C_{new} = c_{new} G$ public key
3. Pick 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}.

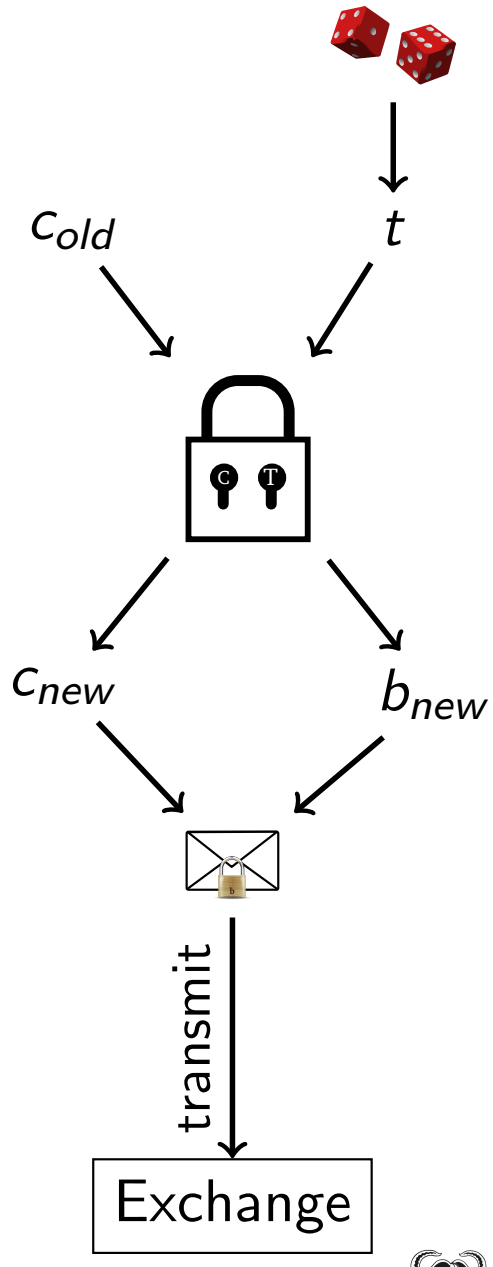


Problem: Owner of c_{new} may differ from owner of c_{old}!

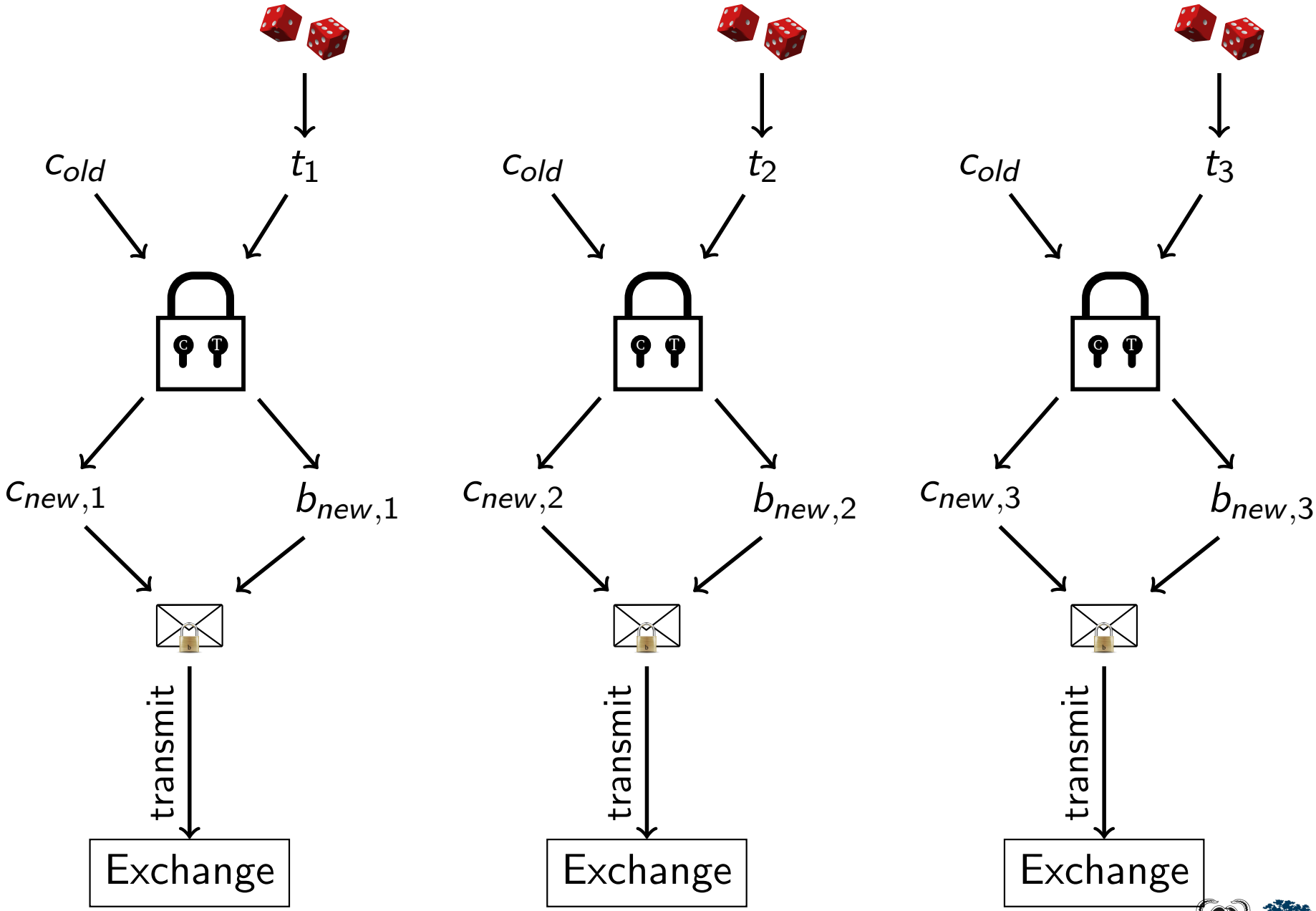
Customer: Transfer key setup (ECDH)

Given partially spent private coin key c_{old}:

1. Let $C_{old} := c_{old} G$ (as before)
2. Create 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



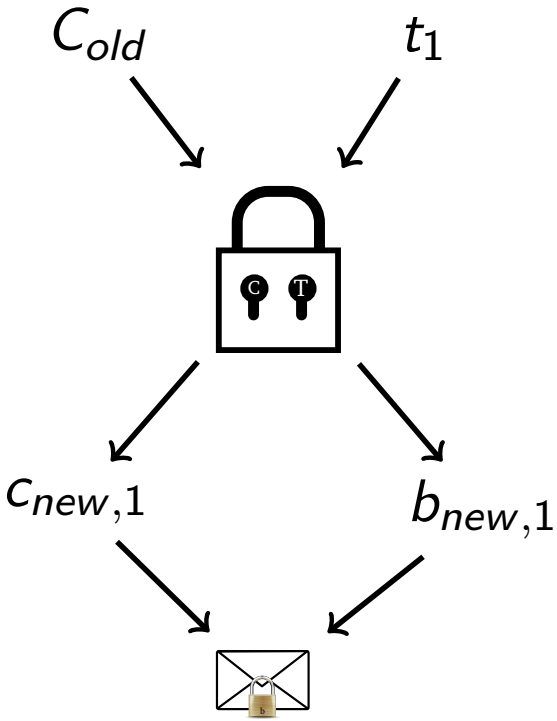
Exchange: Choose!

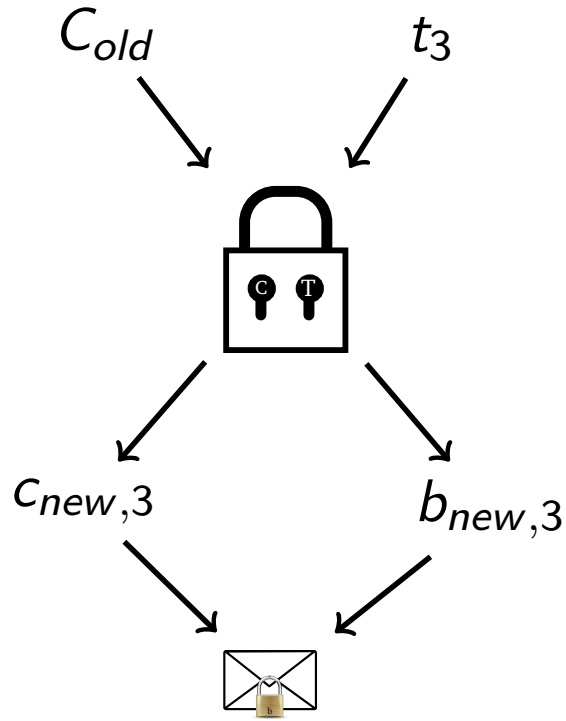
Exchange sends back random $\gamma \in \{1, 2, 3\}$ to the customer.

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

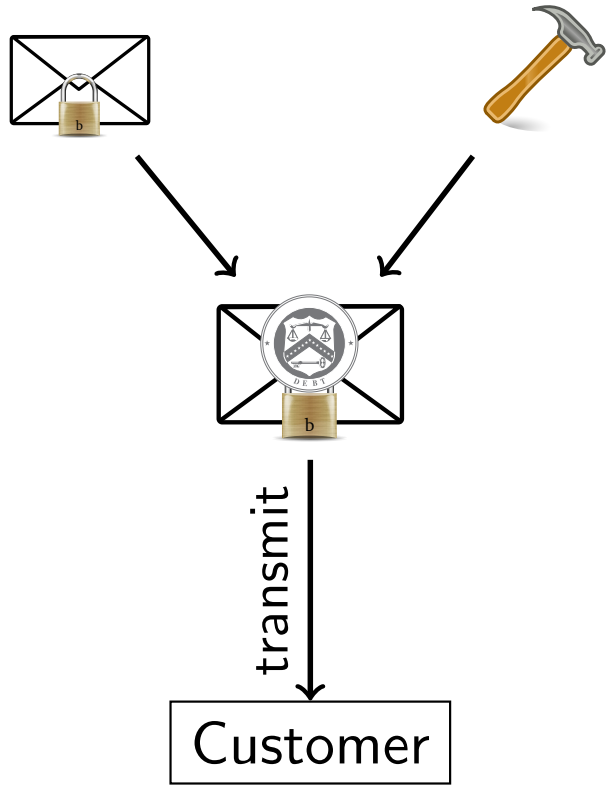
Exchange: Verify ($\gamma = 2$)





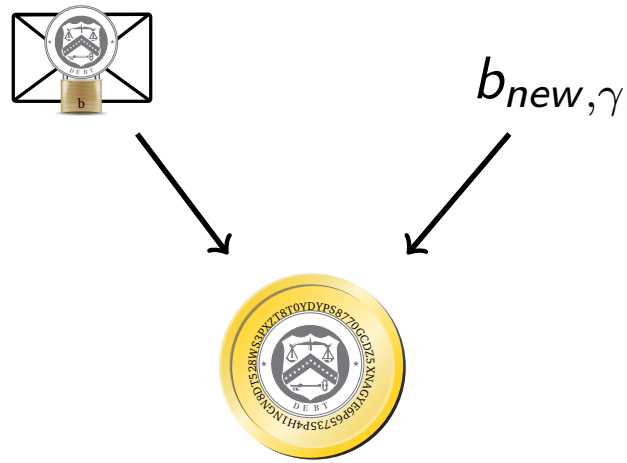
Exchange: Blind sign change (RSA)

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



Customer: Unblind change (RSA)

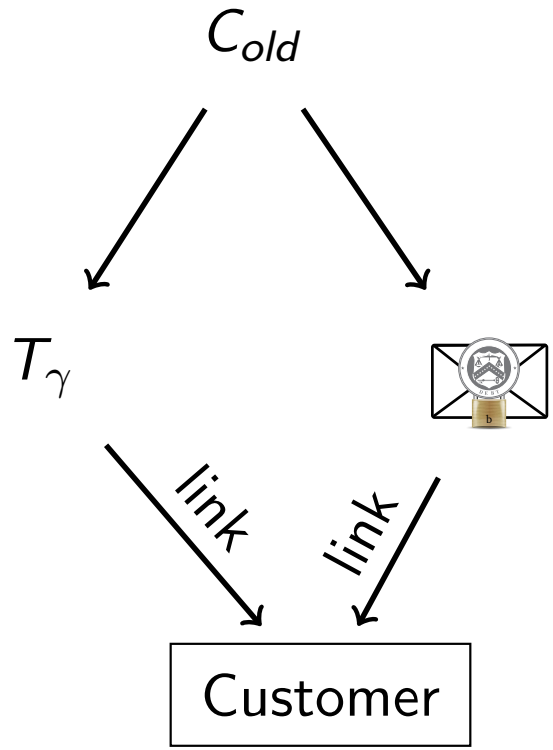
1. Receive s'.
2. Compute $s := s' b_{new,\gamma}^{-1} \mod n$.



Exchange: Allow linking change

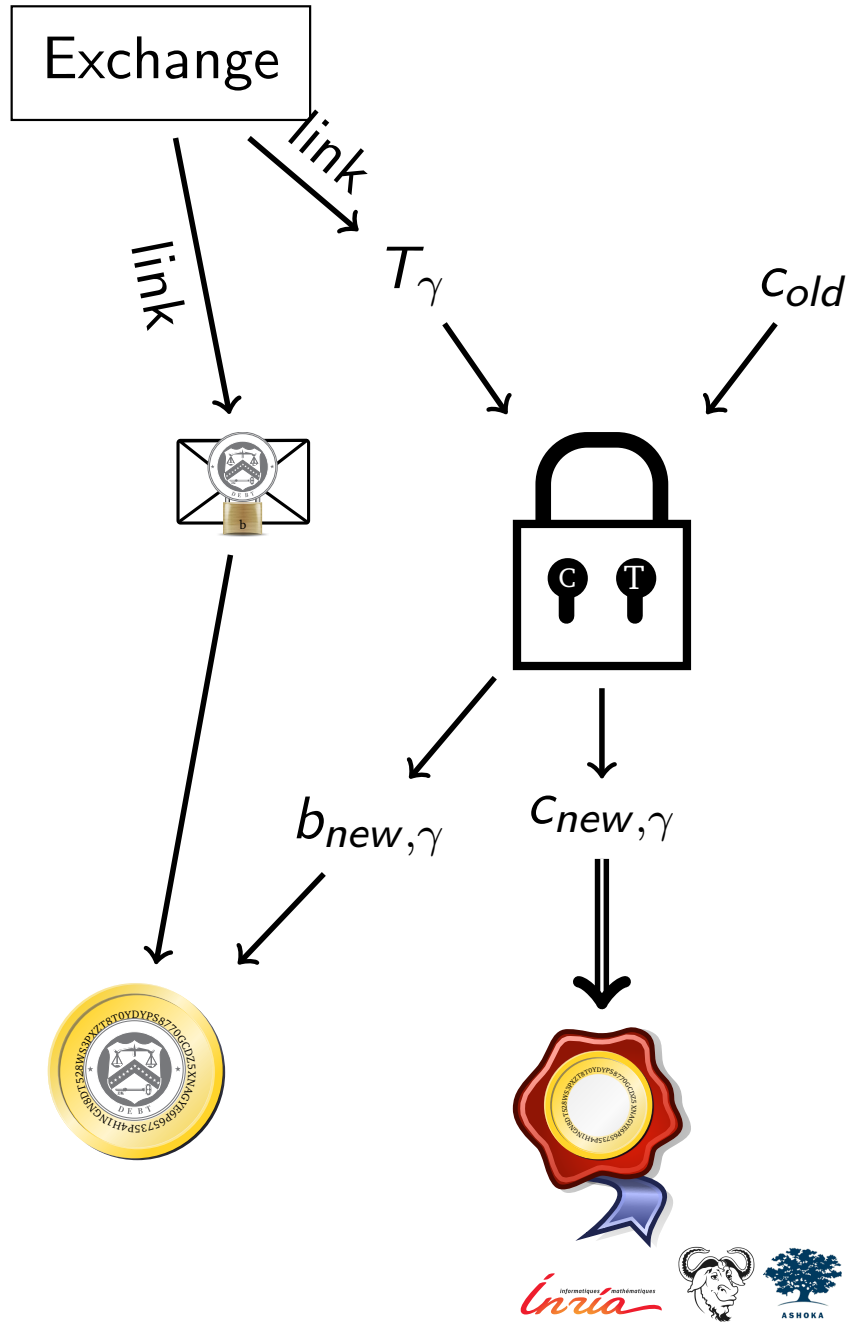
Given C_{old}

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.

Warranting deposit safety

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

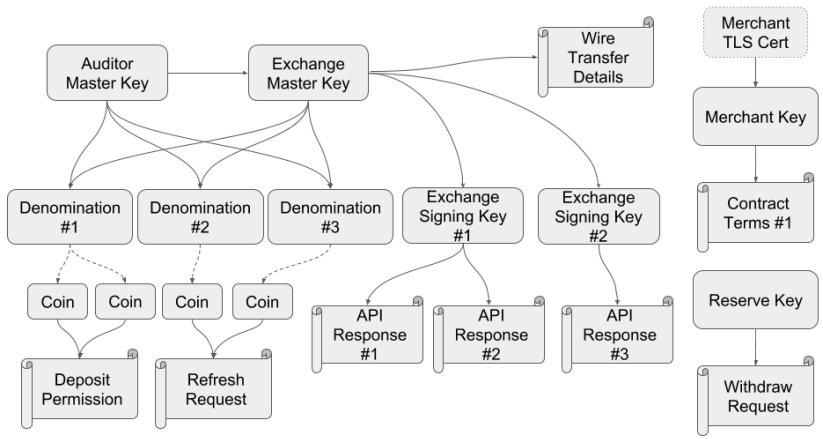
Sends $EdDSA_w(M, H(D), FDH(C))$ to the merchant.

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!

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).

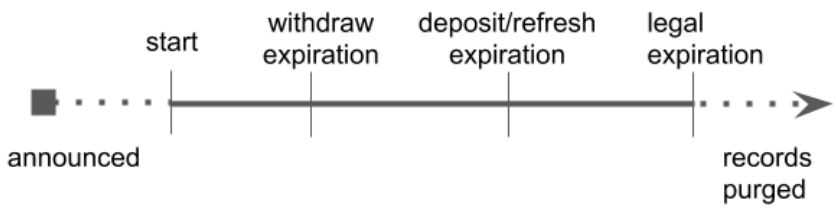


What happens if those private keys are compromised?

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.



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.

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: Project Status

https://docs.taler.net/

- Cryptographic protocols and core exchange component are stable
- Current focus: Merchant integration, settlement integration, wallet backup
- Pilot project at Bern University of Applied Sciences cafeteria
- Internal alpha deployment with a commercial bank in progress

Next Steps: Possible Projects and Collaborations

Area I: System Integration and Partnerships

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

Pilots with banking organizations could:

- Study integration with the underlying RTGS layer:
  - Develop standardized operational procedures
  - Assess transaction performance at scale
  - Perform cost analysis in banking environment
  - Assess effort for integration with commercial banks
- Analyze regulatory considerations for different legislations
- Perform independent security audits of Taler components
- Determine and possibly close gaps in the existing solution

Area II: Development/Research Extensions

Background: https://myoralvillage.org/

We have ideas for protocol extensions and "programmable money":

- Mediated wallet-to-wallet payments (instead of customer-to-merchant)
- Privacy-preserving auctions (trading, currency exchange)
- Age-restricted private payments for children (youth protection)

Central banks should also consider funding research to improve:

- General digital wallet usability and availability
- Accessibility features for illiterate and innumerate users
- Projects that facilitate integration at retailers
  - Hardware and software support for embedded systems
  - Integration into off-the-self E-commerce systems
- Protocol extensions for automated tax reporting

How to support?

Join:	https://lists.gnu.org/mailman/listinfo/taler, irc://irc.freenode.net/#taler
Develop:	https://bugs.taler.net/, https://git.taler.net/
Translate:	https://weblate.taler.net/, translation-volunteer@taler.net
Integrate:	https://docs.taler.net/
Donate:	https://gnunet.org/ev
Invest:	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!

Do you have any questions?

References:

1. David Chaum, Christian Grothoff and Thomas Moser. *How to issue a central bank digital currency*. **SNB Working Papers, 2021**.
2. 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.
3. Jeffrey Burdges, Florian Dold, Christian Grothoff and Marcello Stanisci. *Enabling Secure Web Payments with GNU Taler*. **SPACE 2016**.
4. Florian Dold, Sree Harsha Totakura, Benedikt Müller, Jeffrey Burdges and Christian Grothoff. *Taler: Taxable Anonymous Libre Electronic Reserves*. Available upon request. 2016.
5. 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**.
6. David Chaum, Amos Fiat and Moni Naor. *Untraceable electronic cash*. **Proceedings on Advances in Cryptology, 1990**.
7. Phillip Rogaway. *The Moral Character of Cryptographic Work*. **Asiacrypt**, 2015.

Let money facilitate trade; but ensure capital serves society.

Part II: Integration with the core banking system


High-level Deployment Recipe

. . . as a bank

1. Create an escrow bank account for the exchange with EBICS access
2. Provision offline signing machine (or account during testing)
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 the 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:

- a self-contained play money demo bank
- LibEuFin, an adapter to EBICS and other protocols

LibEuFin

LibEuFin is a standalone project that provides adapters to bank account access APIs.

- LibEuFin provides both a generic access layer and an implementation of the Taler Wire Gateway API for the exchange
- currently, only EBICS 2.5 is supported
- other APIs such as FinTS or PSD2-style XS2A APIs can be added without requiring changes to the Exchange
- tested with a GLS business account

LibEuFin Concepts

- A LibEuFin *bank connection* is a set of credentials and parameters to talk to the bank's account access API.
- A LibEuFin *bank account* is the information about a bank account (balances, transactions, payment initiations) stored locally within the LibEuFin service. A LibEuFin bank account has a default Bank Connection that is used to communicate with the bank's API.
- A *facade* provides a domain-specific access layer to bank accounts and connections. The *Taler Wire Gateway Facade* implements the API required by the Taler exchange and translates it to operations on the underlying account/connection.

LibEuFin Tooling

- `libeufin-nexus` is the main service
- Almost all configuration (except DB credentials) is stored in the database and managed via a RESTful HTTP API
- `libeufin-sandbox` implements a toy EBICS host for protocol testing
- `libeufin-cli` is client for the HTTP API (only implements a subset of available functionality)

LibEuFin Setup Overview

- Obtain EBICS subscriber configuration (host URL, host ID, user ID, partner ID) for the Exchange's escrow account
- Deploy the LibEuFin Nexus service
- Create a new LibEuFin bank connection (of type `ebics`)
- Export and back up the key material for the bank connection (contains EBICS subscriber configuration and private keys)
- Send subscriber initialization to the EBICS host (electronically)
- Export key letter and activate subscriber in the EBICS host (manually)
- Synchronize the bank connection
- Import the account into LibEuFin
- Create a Taler Wire Gateway facade
- Set up scheduled tasks for ingesting new transactions / sending payment initiations

LibEuFin Implementation Limitations

- LibEuFin is less stable than other Taler components, and future updates might contain breaking changes (tooling, APIs and database schema)
- Error handling and recovery is still rather primitive
- The Taler Wire Gateway does not yet implement automatic return transactions when transactions with a malformed subject (i.e. no reserve public key) are received

LibEuFin EBICS Limitations

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

- SEPA Instant Credit Transfers aren't supported yet
- 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)

LibEuFin Setup Guide

https://docs.taler.net/libeufin/nexus-tutorial.html


Part III: Operator security considerations


Key management

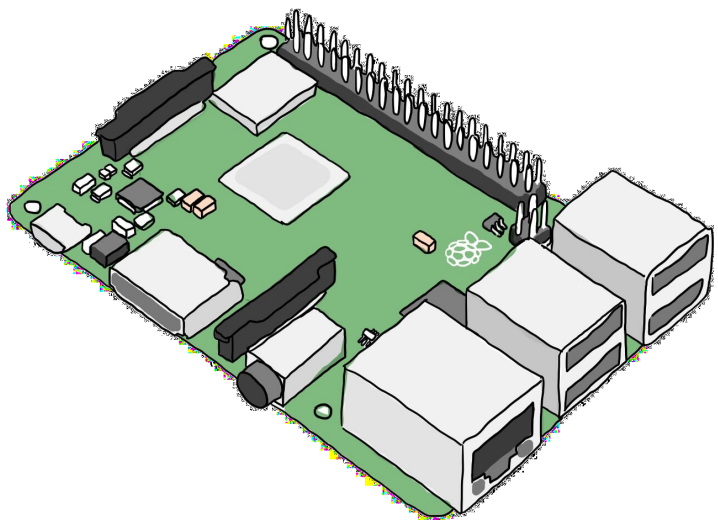
Taler has many types of keys:

- 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.

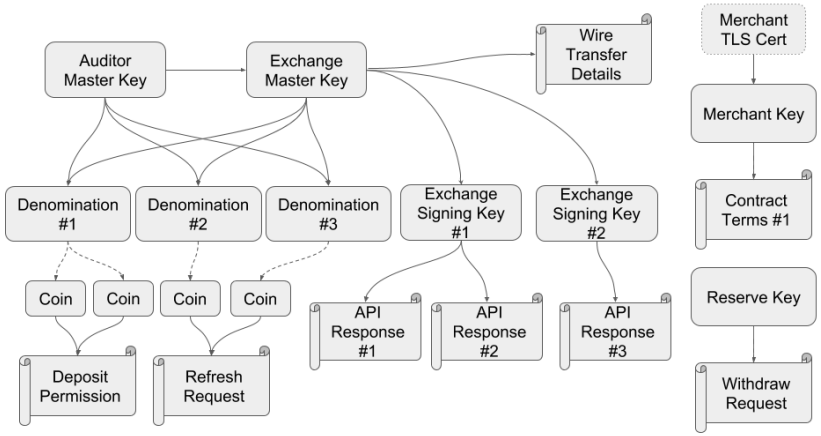
- Those keys must be backed up and remain highly confidential!
- We recommend that computers that have ever had access to those keys to NEVER again go online.
- We recommend using a Raspberry Pi for offline key operations. Store it in a safe under multiple locks and keys.
- Apply full-disk encryption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



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!

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 also not become public.
- The auditor must have a (current) copy. Asynchronous replication is considered sufficient. This copy could also be used 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`.

Nexus

The Nexus has to be able to interact with the escrow account of the bank.

- 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`.

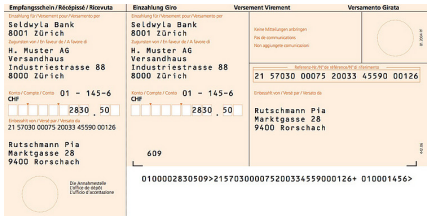
Part IV: Integration considerations

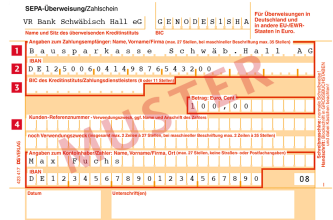
RFC 8905: `payto:` Uniform Identifiers for Payments and Accounts

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.





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!