GNU Taler

Christian Grothoff

Berner Fachhochschule

10.6.2022
Learning Objectives

- Real world surveillance
- Introduction to GNU Taler
- Blind Signatures
- Taler Cryptography
- Integration with the core banking system
- Operator security considerations
- Integration considerations
- Performance
- Depolymerization
- Outlook
- References
What domain of digital communication should we be most concerned about?
Surveillance concerns

- Everybody knows about Internet surveillance.
- But is it that bad?
Surveillance concerns

- Everybody knows about Internet surveillance.
- 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
Where is it worse?

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)
Where is it worse?

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 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)
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
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
Do you want to live under total surveillance?
The Bank of International Settlements
The Emergency Act of Canada\textsuperscript{1}

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

\textsuperscript{1}Speech by Premier Kenney, Alberta, February 2022
Part I: Introduction to GNU 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
- *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”
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
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
Design goals for the GNU Taler Payment System

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

Exchange

Customer

Merchant

Auditor

withdraw coins

verify

deposit coins

spend coins
Architecture of Taler

Exchange’s Bank

Customer’s Bank

Merchant’s Bank

Database

Fees

1. Pay exchange

2. Wire transfer

3. Withdraw coins

4. Spend coins

5. Deposit coins

6. Wire transfer

7. View balance

Browser/Mobile Wallet

Customer

Webshop

Merchant
Usability of Taler

https://demo.taler.net/

1. Install Web extension.
2. Visit the bank.demo.taler.net to withdraw coins.
3. Visit the shop.demo.taler.net to spend coins.
Example: The Taler Snack Machine

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

\[\text{Taler} \text{ Backend}\]

\[\text{Rest API} \quad \text{MDB/ICP} \quad \text{USB} \quad \text{NFC} \quad \text{Wallet}\]

\(^2\)By M. Boss and D. Hofer
Software architecture for the Taler Snack Machine

Application

gnu:net  libnfc  libqrencode  <Taler>

Raspbian

Raspberry Pi

MDB  NFC  TFT

TCP/IP
User story: Install App on Android

https://wallet.taler.net/
User story: Withdraw e-cash
User story: Use machine!
CBDC Initiatives and Taler

Many initiatives are currently at the level of requirements discussion:
▶ ECB: Report on a Digital Euro / Eurosysterm report on the public consultation on a Digital Euro
▶ Bank of England: Just initiated a task force

Taler can serve as the foundation for a bearer-based retail CBDC.
▶ Taler replicates physical cash rather than bank deposits
▶ Taler has unique design principles and regulatory features that align with CBDC requirements
▶ ECB survey has identified privacy as a primary requirement of end users
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
Offline capabilities are often cited as a requirement for CBDC. All implementations must either use restrictive hardware elements and/or introduce counterparty risk.

- Permanent offline features weaken a CBDC solution (privacy, security).
- Introduces unwarranted competition for physical cash (endangers emergency-preparedness).

We recommend a tiered approach:

1. Online-first, bearer-based CBDC
2. (Optional:) Limited offline mode for network outages
3. Physical cash for emergencies (power outage, catastrophic cyber incidents)
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
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
Break
Reminder: RSA

Pick $p, q$ prime and $e$ such that

$$GCD((p - 1)(q - 1), e) = 1$$

(1)

- Define $n = pq$,
- compute $d$ such that $ed \equiv 1 \mod (p - 1)(q - 1)$.
- Let $s := m^d \mod n$.
- Then $m \equiv s^e \mod n$. 
RSA Summary

- Public key: \( n, e \)
- Private key: \( d \equiv e^{-1} \mod \phi(n) \) where \( \phi(n) = (p - 1) \cdot (q - 1) \)
- Encryption: \( c \equiv m^e \mod n \)
- Decryption: \( m \equiv c^d \mod n \)
- Signing: \( s \equiv m^d \mod n \)
- Verifying: \( m \equiv s^e \mod n \)?
Low Encryption Exponent Attack

- $e$ is known
- $M$ maybe small
- $C = M^e < n$?
- If so, can compute $M = \sqrt[e]{C}$
  ⇒ Small $e$ can be bad!
Padding and RSA Symmetry

- Padding can be used to avoid low exponent issues (and issues with \( m = 0 \) or \( m = 1 \))
- Randomized padding defeats chosen plaintext attacks
- Padding breaks RSA symmetry:
  \[
  D_{A_{\text{priv}}} (D_{B_{\text{priv}}} (E_{A_{\text{pub}}} (E_{B_{\text{pub}}} (M)))) \neq M
  \]  
- PKCS#1 / RFC 3447 define a padding standard
Blind signatures with RSA [2]

1. Obtain public key
   \((e, n)\)

2. Compute
   \(f := FDH(m),\)
   \(f < n.\)

3. Pick blinding factor
   \(b \in \mathbb{Z}_n\)

4. Transmit
   \(f' := fb^e \mod n\)
Blind signatures with RSA [2]

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

1. Receive \(f'.\)
2. Compute 
   \(s' := f'^d \mod n.\)
3. Send \(s'.\)
Blind signatures with RSA [2]

1. **Obtain public key**
   
   $(e, n)$

2. **Compute**
   
   $f := FDH(m)$,
   
   $f < n$.

3. **Pick blinding factor**
   
   $b \in \mathbb{Z}_n$

4. **Transmit**
   
   $f' := fb^e \ mod \ n$

1. **Receive** $f'$.

2. **Compute**
   
   $s' := f'^d \ mod \ n$.

3. **Send** $s'$.

1. **Receive** $s'$.

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

Capability: $m \Rightarrow M$
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 \(f' := fb^e \mod n\)

\[
XNAGYE6P65735P4H1NGN8DT528W
S 3 P X Z T 8 T 0 Y D Y P S 8 7 7 0 G C D Z5
\]
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

Taler (Withdraw coins)

Customer Browser

Bank Site

Taler Exchange

HTTPS

HTTPS

wire transfer

1 user authentication

2 send account portal

3 initiate withdrawal (specify amount and exchange)

4 request coin denomination keys and wire transfer data

5 send coin denomination keys and wire transfer data

6 execute withdrawal

opt

7 request transaction authorization

8 transaction authorization

9 withdrawal confirmation

10 execute wire transfer

11 withdraw request

12 signed blinded coins

13 unblind coins

Customer Browser

Bank Site

Taler Exchange
Customer: Build shopping cart

www

WWW

transmit

Merchant
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)

\[ s^e \equiv FDH(C) \mod n \]
Payment processing with Taler

1. Choose goods by navigating to offer URL
2. Send signed digital contract proposal
3. Select Taler payment method (skippable with auto-detection)
4. Affirm contract
5. Navigate to fulfillment URL
6. Send hash of digital contract and payment information
7. Send payment
8. Forward payment
9. Confirm payment
10. Confirm payment
11. Reload fulfillment URL for delivery
12. Provide product resource
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 \(d\).

⇒ 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.
Break
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
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 $f'_{new} := f_{new}b_{new}^e \mod n$

... 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 $f'_{new} := f_{new}b_{new}^e \mod n$
   ... 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 $f'_{new} := f_{new} b_{new}$
Cut-and-Choose

\[ \begin{align*}
&c_{old} \\
&\downarrow \\
&c_{new,1} \\
&\downarrow \\
&\text{Exchange}
\end{align*} \]

\[ \begin{align*}
&t_1 \\
&\downarrow \\
&b_{new,1} \\
&\downarrow \\
&\text{Exchange}
\end{align*} \]

\[ \begin{align*}
&c_{old} \\
&\downarrow \\
&c_{new,2} \\
&\downarrow \\
&\text{Exchange}
\end{align*} \]

\[ \begin{align*}
&t_2 \\
&\downarrow \\
&b_{new,2} \\
&\downarrow \\
&\text{Exchange}
\end{align*} \]

\[ \begin{align*}
&c_{old} \\
&\downarrow \\
&c_{new,3} \\
&\downarrow \\
&\text{Exchange}
\end{align*} \]

\[ \begin{align*}
&t_3 \\
&\downarrow \\
&b_{new,3} \\
&\downarrow \\
&\text{Exchange}
\end{align*} \]
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 $s' := f'_{new, \gamma}^d \mod n$.
3. Send signature $s'$.
Customer: Unblind change (RSA)

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

$XNAGYE6P65735P4H1NGN8DT528W$

$S3PXZT8T0YDYPS8770GCDZ5$
Exchange: Allow linking change

Given $C_{old}$

return $T_{\gamma}$ and

$s := s' b_{new,\gamma}^{-1} \mod n.$
1. Have $c_{old}$.
2. Obtain $T_\gamma$, $s$ from exchange
3. Compute $X_\gamma = c_{old} T_\gamma$
4. Derive $c_{new,\gamma}$ and $b_{new,\gamma}$ from $X_\gamma$
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.
Part II: Integration with the core banking system
Taler: Bank Perspective

---

- **Nginx**
  - **REST API**
    - **Exchange**
      - **Internal REST API**
        - **Postgres**
          - **SQL**
            - **Core Banking**
            - **EBICS/FinTS**
              - **Nexus**
                - **SQL**
                  - **Postgres**
Taler: Auditor Perspective

- `auditor-httpd`
- `auditor-report`

Sync:

- `Postgres (Auditor)`
- `Postgres (Bank)`
Taler: Merchant Perspective

- E-commerce Frontend
- Backoffice

Diagram:
- E-commerce Frontend to `taler-merchant-httpd` via REST API
- `taler-merchant-httpd` to Postgres via SQL
- `taler-merchant-httpd` to Sqlite via SQL
- `taler-merchant-httpd` to Backoffice via REST API

Other databases:
- ...
Taler: Wallet Architecture

Background: https://anastasis.lu/
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"
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`, `taler-exchange-secmod-cs` 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.
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.

⇒ Configuration should be separate from `taler-exchange-httpd`. 
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.
- Configuration should be separate from `taler-exchange-httpd`. 
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](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!
Part V: Performance
Performance

Other Payment Systems

Bitcoin

? TPS
Performance
Other Payment Systems

Bitcoin

4 TPS
Performance

Other Payment Systems

Bitcoin

4 TPS
### Performance

#### Other Payment Systems

<table>
<thead>
<tr>
<th>Payment System</th>
<th>TPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitcoin</td>
<td>4</td>
</tr>
<tr>
<td>PayPal</td>
<td>193</td>
</tr>
</tbody>
</table>
Performance
Other Payment Systems

Bitcoin
4 TPS

PayPal
193 TPS

[06.22] - Researchgate
Performance
Other Payment Systems

<table>
<thead>
<tr>
<th>Payment System</th>
<th>TPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitcoin</td>
<td>4 TPS</td>
</tr>
<tr>
<td>PayPal</td>
<td>193 TPS</td>
</tr>
<tr>
<td>Visa</td>
<td>1’667 TPS</td>
</tr>
</tbody>
</table>

[06.22] - Researchgate
## Performance

### Other Payment Systems

<table>
<thead>
<tr>
<th>Payment System</th>
<th>TPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitcoin</td>
<td>4</td>
</tr>
<tr>
<td>PayPal</td>
<td>193</td>
</tr>
<tr>
<td>Visa</td>
<td>1,667</td>
</tr>
</tbody>
</table>

[06.22] - Researchgate
Performance
CBDC Projects

e-Krona (Sweden)

100 TPS
Performance

CBDC Projects

e-Krona (Sweden)

100 TPS
Performance

CBDC Projects

e-Krona (Sweden)  e-CNY (China)

100 TPS  10’000 TPS
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
## Performance

### CBDC Projects

<table>
<thead>
<tr>
<th>Project</th>
<th>Country/Traits</th>
<th>TPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>e-Krona (Sweden)</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>e-CNY (China)</td>
<td></td>
<td>10,000</td>
</tr>
<tr>
<td>Project Hamilton (MIT)</td>
<td></td>
<td>1,700,000</td>
</tr>
</tbody>
</table>

[06.22] - BostonFed - Atlantic 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
Allocate an Experiment

1. Build Image (Kameleon)
Allocate an Experiment

1. Build Image (Kameleon)
2. Copy Image to Grid'5000
Allocate an Experiment

1. Build Image (Kameleon)
2. Copy Image to Grid'5000
3. Allocate Experiment (jFed)
Allocate an Experiment

1. Build Image (Kameleon)
2. Copy Image to Grid'5000
3. Allocate Experiment (jFed)
Horizontal Distribution
Dashboard
Part VI: Depolymerization\textsuperscript{6}

\textsuperscript{6}By Antoine d'Aligny
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
Architecture

Auditor

Settlement Layer

Deposit money ➔ Withdraw money

Taler payment system

Deposit coins ➔ Withdraw coins

Customer ➔ Merchant

Spend coins

Settlement layer

- This work, Blockchain!

Taler payment system

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

- **Wire Gateway API**
  - **Taler Exchange**
    - HTTP
  - **Wire Gateway**
    - SQL
  - **PostgreSQL**
    - SQL
- **DLT specific**
  - **DLT Full Node**
    - RPC
  - **DLT Adapter**

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

- Wait for notifications
- Reconcile local DB with DLT
- Trigger debits
- Reissue stuck debits
- Bounce malformed credits

**Figure:** Worker loop

**DLT reconciliation**
- 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
Part VII: Outlook
Summary

- We can design protocols that fail soft.
- GNU Taler’s design limits financial damage even in the case private keys are compromised.
- GNU Taler does:
  - Gives change, can provide refunds
  - Integrates nicely with HTTP, handles network failures
  - High performance
  - Free Software
  - Formal security proofs
GNU Taler: Current Work

Ongoing work:

- Wallet-to-wallet payments
- Payments with zero-knowledge age verification
- Internationalization ⇒ https://weblate.taler.net/
Bachelor Thesis topics

- Address remaining scalability challenges (multiple topics)
- Porting to more platforms (Web shops, iOS, embedded)
- Integration of P2P payments (e-mail, SMS, twitter, Signal, etc.)
- Implement currency conversion service
- Improve design and usability for illiterate and innumerate users
- Integration with KYC/AML providers
- SAP integration of BFH snack machine with BFH SAP
- Federated exchange (wads)
Visions

- Be paid to read advertising, starting with spam
- Give welfare without intermediaries taking huge cuts
- Forster regional trade via regional currencies
- Eliminate corruption by making all income visible
- Stop the mining by making crypto-currencies useless for anything but crime
References

Jeffrey Burdges, Florian Dold, Christian Grothoff, and Marcello Stanisci. 
Enabling secure web payments with GNU Taler. 

David Chaum. 

How to issue a central bank digital currency. 