GNU Taler: Ein neues elektronisches Bezahlsystem

Christian Grothoff

Institut National de Recherche en Informatique et en Automatique (Inria)
The GNU Project
Ashoka Fellow

30.11.2016

“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.”

Motivation

Modern economies need currency ...
Modern economies need payments online.
SWIFT/Mastercard/Visa are too transparent.
Bitcoin

- Unregulated payment system and currency:
  ⇒ lack of regulation is a feature!
- Decentralised peer-to-peer system
Bitcoin

- Unregulated payment system and currency:
  ⇒ lack of regulation is a feature!
- Decentralised peer-to-peer system
- Decentralised banking requires solving Byzantine consensus
- Creative solution: tie initial accumulation to solving consensus
Unregulated payment system and currency: lack of regulation is a feature!

Decentralised peer-to-peer system

Decentralised banking requires solving Byzantine consensus

Creative solution: tie initial accumulation to solving consensus

Proof-of-work advances ledger

Very expensive banking
Current average transaction value: \( \approx 1000 \text{ USD} \)
Cryptography is rather primitive:

**All Bitcoin transactions are public and linkable!**

⇒ no privacy guarantees
⇒ enhanced with “laundering” services

ZeroCoin, CryptoNote (Monero) and ZeroCash (ZCoin) offer anonymity.
Is society ready for an anarchistic economy?
GNU Taler

Digital cash, made socially responsible.

Taxable, Anonymous, Libre, Practical, Resource Friendly
Architecture of GNU Taler

Customer → Exchange  
withdraw coins → verify

Exchange → Merchant  
spend coins → deposit coins

Auditor
Usability of Taler

https://demo.taler.net/

1. Install Chrome extension.
2. Visit the bank.demo.taler.net to withdraw coins.
3. Visit the shop.demo.taler.net to spend coins.
Value proposition: Customer

- Convenient: pay with one click
- Guaranteed: never fear being rejected by false-positives in the fraud detection
- Secure: like cash, except no worries about counterfeit
- Privacy-preserving: payment requires no personal information
- Stable: no currency fluctuations, pay in traditional currencies
- Free software: no hidden “gadgets”, third parties can verify
Value proposition: Merchant

- Fast: transactions at Web-speed
- Secure: signed contracts, no legitimate customer rejected by fraud deception
- Free software: competitive pricing and support
- Low fees: efficient protocol + no fraud = low costs
- Flexible: any currency, any amount
- Ethical: no fluctuation risk, no pyramid scheme, not suitable for illegal business
- Legal: complies with Regulation (EU) 2016/679 (GDPR)\(^1\)

\(^1\)Requires privacy by design and data minimization for all data processing in Europe after 25.5.2018.
Value proposition: Government

- Free software = commons: no monopoly, preserve independence
- Efficiency: high transaction costs hurt the economy
- Security: signed contracts, no counterfeit
- Audited: no bad banks
- Privacy: protection against foreign espionage
- Taxability: reduces black markets
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.
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
Merchant Integration: Wallet Detection

<script src="taler-wallet-lib.js"></script>
<script>
    taler.onPresent(() => {
        alert("Taler wallet is installed");
    });
    taler.onAbsent(() => {
        alert("Taler wallet is not installed");
    });
</script>
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":"YTHOC4QBCQ10VDNTJN0DCTTV2Z6JHT5NF43F0RQHZ8JYB5NG4W4G...",  
"amount":{"currency":"EUR","fraction":1,"value":0},  
"auditors": [{"auditor_pub":"42V6TH91Q83FB846DK1GW3JQ5E8DS273W4..."}],  
"exchanges": [{"master_pub":"1T5FA8VQHMMKBHDMDYPRA2ZFK2S63AKFOY..."},  
  
"expiry":"/Date(1480119270)/",  
"fulfillment_url": "https://shop/article/42?tid=249&time=14714744",  
"max_fee": {"currency": "EUR", "fraction": 01, "value": 0},  
  
"merchant": {"address": "Mailbox 4242", "jurisdiction": "Jersey",  
  "name": "Shop Inc."},  
"merchant_pub": "Y1ZAR5346J3ZTEXJCHQY9N78EZ2HSKZK8M0MYTNRJG5N...",  
"products": [{  
  "description": "Essay: The GNU Project",  
  "price": {"currency": "EUR", "fraction": 1, "value": 0},  
  "product_id": 42, "quantity": 1}],  
"refund_deadline": "/Date(1471522470)/",  
"timestamp": "/Date(1471479270)/",  
"transaction_id": 249960194066269
}
Information for system integrators

https://api.taler.net/
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.
Global setup: Pick an Elliptic curve

Need:

- \( G \) generator in ECC curve, a point
- \( n \) size of ECC group, \( o := |G|, \) \( o \) prime

Now we can, for example, compute:

- \( A = G + G \)
  - \( = 2G \)
- \( B = A + G \)
  - \( = 3G \)
- \( C = cG \) for \( c \in \mathbb{Z} \)

Note:

- \( G = (o + 1)G \)
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 $(e, n)$. 

\[ (p, q) \]
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 \(m := FDH(C)\), \(m < n\).
3. Pick blinding factor \(b \in \mathbb{Z}_n\)
4. Transmit \(m' := mb^e \mod n\)
Exchange: Blind sign (RSA)

1. Receive $m'$.
2. Compute $s' := m'^d \mod n$.
3. Send signature $s'$. 
Customer: Unblind coin (RSA)

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

\[
\begin{align*}
\text{Envelope} & \quad b \\
\text{Coin} & \quad b^{-1}
\end{align*}
\]
Customer: Build shopping cart
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
Merchant and Exchange: Verify coin (RSA)

\[ s^e \equiv m \mod n \]
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.
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 $m_{new} := FDH(C_{new})$, $m < n$.
5. Transmit $m'_{new} := m_{new}b_{new}^e \mod n$

... 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 $m_{new} := FDH(C_{new}), m < n.$
5. Transmit $m'_{new} := m_{new} b_{new}^e \mod n$

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

Problem: Owner of $c_{new}$ may differ from owner of $c_{old}$!
Diffie-Hellman (ECDH)

1. Create private keys $d, h \mod o$
2. Define $D = dG$
3. Define $H = hG$
4. Compute $DH := d(hG) = h(dG)$
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 $m_{new} := FDH(C_{new})$
8. Transmit $m'_{new} := m_{new} b_{new}^e$
Cut-and-Choose

1. \( c \) transmitted to \( t_1 \)
2. \( c \) transmitted to \( t_2 \)
3. \( c \) transmitted to \( t_3 \)

\( c_{\text{new},1} \) transmitted
\( b_{\text{new},1} \) transmitted

\( c_{\text{new},2} \) transmitted
\( b_{\text{new},2} \) transmitted

\( c_{\text{new},3} \) transmitted
\( b_{\text{new},3} \) transmitted

Exchange
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 $m'_{\text{new}, \gamma}$.
2. Compute $s' := m'^d_{\text{new}, \gamma} \mod n$.
3. Send signature $s'$.
Customer: Unblind change (RSA)

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

\[ b_{new,\gamma} \]
Exchange: Allow linking change

Given $C_{old}$

return $T_{\gamma}, s := s' b_{new, \gamma}^{-1} \mod n.$
Customer: Link (threat!)

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
  ⇒ 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.

Transactions via refresh are equivalent to sharing a wallet.
Current technical developments

- Tutorial for merchants
- Tutorial for Web shop integration
- Improving wallet (error handling, features, browser support)
- Ongoing work on exchange auditing
Business considerations

- Exchange needs to be a legal (!) business to operate.
- Exchange operator income is from *transaction fees*.
- Created Taler Systems S.A. in Luxemburgh.
- Now trying to find partners and financing for startup.
## Payment solutions - Pricing

<table>
<thead>
<tr>
<th>Provider</th>
<th>Pricing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alipay</td>
<td>2.0% - 3.0%</td>
</tr>
<tr>
<td>Allied Wallet</td>
<td>1.95% + $ 0.20</td>
</tr>
<tr>
<td>Amazon Payments</td>
<td>2.9% + $ 0.30</td>
</tr>
<tr>
<td>Avangate</td>
<td>4.9% + $ 2.50</td>
</tr>
<tr>
<td>Billpro</td>
<td>2.1% + 3.5% fee</td>
</tr>
<tr>
<td>BitGold Inc.</td>
<td>1% fee on every purchase</td>
</tr>
<tr>
<td>Checkout.com</td>
<td>2.95% - 3.95% + £0.15</td>
</tr>
<tr>
<td>Coinify (Bitcoin)</td>
<td>0%</td>
</tr>
<tr>
<td>eComCharge</td>
<td>3.5% + €0.35</td>
</tr>
<tr>
<td>GoCardless</td>
<td>1% up to a maximum of £2</td>
</tr>
<tr>
<td>LaterPay (Cookie)</td>
<td>15%</td>
</tr>
<tr>
<td>Western Union</td>
<td>Variable — From 5% up</td>
</tr>
<tr>
<td>Taler Systems S.A. (planned)</td>
<td>0.5%</td>
</tr>
</tbody>
</table>
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:


Let money facilitate trade; but ensure capital serves society.
## Competitor comparison

<table>
<thead>
<tr>
<th></th>
<th>Cash</th>
<th>Bitcoin</th>
<th>Zerocoin</th>
<th>Creditcard</th>
<th>Taler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online</td>
<td>−−−</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+++</td>
</tr>
<tr>
<td>Offline</td>
<td>+++</td>
<td>−−</td>
<td>−−</td>
<td>+</td>
<td>−−</td>
</tr>
<tr>
<td>Trans. cost</td>
<td>+</td>
<td>−−−</td>
<td>−−−</td>
<td>−</td>
<td>++</td>
</tr>
<tr>
<td>Speed</td>
<td>+</td>
<td>−−−</td>
<td>−−−</td>
<td>o</td>
<td>++</td>
</tr>
<tr>
<td>Taxation</td>
<td>−</td>
<td>−−</td>
<td>−−−</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Payer-anon</td>
<td>++</td>
<td>o</td>
<td>++</td>
<td>−−−</td>
<td>+++</td>
</tr>
<tr>
<td>Payee-anon</td>
<td>++</td>
<td>o</td>
<td>++</td>
<td>−−−</td>
<td>−−−</td>
</tr>
<tr>
<td>Security</td>
<td>−</td>
<td>o</td>
<td>o</td>
<td>−</td>
<td>++</td>
</tr>
<tr>
<td>Conversion</td>
<td>+++</td>
<td>−−−</td>
<td>−−−</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Libre</td>
<td>−</td>
<td>+++</td>
<td>+++</td>
<td>−</td>
<td>+++</td>
</tr>
</tbody>
</table>
## Evolution Matrix

<table>
<thead>
<tr>
<th></th>
<th>ZeroCoin</th>
<th>CryptoNote</th>
<th>ZeroCash</th>
<th>GNU Taler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td>45000 b</td>
<td>13000 b</td>
<td>1000 b</td>
<td>≈ 1000 b</td>
</tr>
<tr>
<td>CPU spend</td>
<td>500 ms</td>
<td>10 ms</td>
<td>45s</td>
<td>1 ms</td>
</tr>
<tr>
<td>CPU verify</td>
<td>450 ms</td>
<td>10 ms</td>
<td>6 ms</td>
<td>2 ms</td>
</tr>
<tr>
<td>Anonymity set</td>
<td>ZC subset</td>
<td>freq. dep.</td>
<td>all users</td>
<td>customers</td>
</tr>
<tr>
<td>Change</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Scalability</td>
<td>−−−</td>
<td>?</td>
<td>?</td>
<td>+++</td>
</tr>
<tr>
<td>Trans. cost</td>
<td>100 USD?</td>
<td>20 USD?</td>
<td>20 USD?</td>
<td>0.0001 USD</td>
</tr>
</tbody>
</table>

Note: Approximate figures based on current reading of the papers, not on scientific comparative experiments.
Bitcoin anonymization technology

- Coin creation and distributed ledger basically remain
- Transactions are no longer simply signed by the owner
- Instead, currency is transferred from a pool of owners to another pool of owners
- Impossible to say which specific owner initiated the transfer (there is just a zero knowledge proof it was somebody authorized)
- Impossible to say who became the new owner (just somebody now can prove that he has the right to authorize a transfer)
- Cryptography used is somewhat experimental (zk-SNARK) and expensive
  ⇒ Further research might enable us to find attacks (arms race)
Taler /keys

GET /keys

200 OK: $T(DK, A_{DK}, M), S_{M}(SK)$

- **$T$**: financial regulator key
  - Necessarily pinned
- **$DK$**: RSA public key
  - ("denomination key")
- **$A_{DK}$**: Value of coins signed by $DK$
- **$M$**: Offline master key of exchange
- **$SK$**: Online signing key of exchange
Taler /withdraw/sign

Result: \( \langle c, S_{DK}(C) \rangle \).

\begin{align*}
A & \text{ Some amount, } A \geq A_{DK} \\
RK & \text{ Reserve key} \\
DK & \text{ Denomination key} \\
b & \text{ Blinding factor} \\
B_b() & \text{ RSA-FDH blinding} \\
C & \text{ Coin public key } C := cG \\
S_{RK}() & \text{ EdDSA signature} \\
S_{DK}() & \text{ RSA-FDH signature}
\end{align*}
Taler /deposit

Merchant and exchange see only the public coin $\langle C, S_{DK}(C) \rangle$.

\[ \begin{align*}
\text{POST } /\text{deposit } & S_{DK}(C).S_c(D) \\
\text{200 OK: } & S_{SK}(S_c(D)) \\
\text{409 CONFLICT: } & S_c(D')
\end{align*} \]

- $DK$  Denomination key
- $S_{DK}()$ RSA-FDH signature using $DK$
- $c$  Private coin key, $C := cG$.
- $S_C()$ EdDSA signature using $c$
- $D$  Deposit details
- $SK$  Exchange’s signing key
- $S_{SK}()$ EdDSA signature using $SK$
- $D'$  Conflicting deposit details $D' \neq D$
\( \kappa \) System-wide security parameter, usually 3.

\( \mathcal{DK} := \left[ DK(i) \right]_i \)
List of denomination keys
\( D + \sum_i A_{\mathcal{DK}(i)} < A_{\mathcal{DK}} \)

\( t_j \) Random scalar for \( j < \kappa \)

\( T := [T_j]_\kappa \) where \( T_j = t_j G \)

\( k_j := cT_j = t_j C \) is an ECDHE

\( b_{j(i)} := KDF_b(k_j, i) \)

\( c_{j(i)} := KDF_c(k_j, i) \)

\( C_{j(i)} := c_{j(i)} G \)

\( B := [H(\beta_j)]_\kappa \) where
\( \beta_j := \left[ B_{b_{j(i)}}(C_{j(i)}) \right]_i \)

\( \gamma \) Random value in \([0, \kappa)\)
Taler /refresh/reveal

\[ \begin{align*}
\mathcal{DK} & := [DK^{(i)}]_i \\
T & := [t_j | j \in \kappa, j \neq \gamma] \\
k_\gamma & := cT_\gamma = t_\gamma C \\
b_\gamma^{(i)} & := KDF_b(k_\gamma, i) \\
c_\gamma^{(i)} & := KDF_c(k_\gamma, i) \\
C_\gamma^{(i)} & := c_\gamma^{(i)} G \\
B_\gamma^{(i)} & := B_{b_\gamma^{(i)}}(C_\gamma^{(i)}) \\
\beta_\gamma & := [B_\gamma^{(i)}]_i \\
S & := [S_{\mathcal{DK}(i)}(B_\gamma^{(i)})]_i \\
Z & \text{ Cut-and-choose mismatch information}
\end{align*} \]
Taler /refresh/link

POST /refresh/link C

200 OK: $T_\gamma$

404 NOT FOUND

$C$ Old coind public key

$T_\gamma$ Linkage data $\mathcal{L}$ at $\gamma$