The Bank’s Online Payment 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 Online Payment Problem

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

- Unregulated payment system and currency:
  ⇒ lack of regulation is a feature!
- Implemented in free software
- Decentralised peer-to-peer system
The Distraction: Bitcoin

- Unregulated payment system and currency:
  ⇒ 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:
  ⇒ 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
  ⇒ Proof-of-work advances ledger
  ⇒ Very expensive banking
Average transaction value: $4215$ USD (on 9.8.2018)
Cryptography is rather primitive:

All Bitcoin transactions are public and linkable!

⇒ no privacy guarantees
⇒ 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
Taler Overview

- Exchange
  - withdraw coins
  - deposit coins
  - spend coins
- Auditor
  - verify
- Customer
- Merchant
Architecture of Taler

1. pay exchange
2. wire transfer
3. withdraw coins
4. spend coins
5. deposit coins
6. wire transfer
7. view balance

⇒ Convenient, taxable, privacy-enhancing, & resource friendly!
Usability of Taler

1. Install Browser extension.
2. Visit the bank.demo.taler.net to withdraw coins.
3. Visit the shop.demo.taler.net to spend coins.
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 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 Case: Anti-Spam

Today, $\text{p} \equiv \text{p}$ 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
Use Case: Anti-Spam

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
Social Impact of Taler

- for the disadvantaged
- for ordinary citizens
- for a better Internet
- for a better market economy
- for new economies

- anti-discrimination
- financial education
- accessibility
- regional markets
- alternative economies
- economic independence
- economic integration (migrants)

- libre
- anti-corruption
- green / efficient
- comfort
- privacy
- anti-DDoS
- Internet security
- anti-spam

- supports affordability
- improves competition
- for ordinary citizens
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.
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: $c \Rightarrow$
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

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
7. Opt: Request transaction authorization
8. Request transaction authorization
9. Withdrawal confirmation
10. Execute wire transfer
11. Withdraw request
12. Signed blinded coins
13. Unblind coins
Customer: Build shopping cart
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":0,"value":1},
  "max_fee":{"currency":"EUR","fraction":100000,"value":0},
  "auditors":[{"auditor_pub":"42V6TH91Q83FB846DK1GW3JQ5E8DS273W4..."}]
  "exchanges":[{"master_pub":"1T5FA8VQHMMKBHDMYPRZA2ZFK2S63AKFOY..."},
    "url":"https://exchange/"],
  "fulfillment_url": "https://shop/article/42?tid=249&time=14714744",
  "merchant":{"address":"Mailbox 4242","jurisdiction":"Jersey",
    "name":"Shop Inc."},
  "merchant_pub":"Y1ZAR5346J3ZTEXJCHQY9N78EZ2HSKZK8M0MYTNRJG5N...",
  "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)

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

**Request Offer**
1. Choose goods by navigating to offer URL
2. Send signed digital contract proposal
3. Select Taler payment method (skippable with auto-detection)

**Execute Payment**
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

**Fulfillment**
10. Confirm payment
11. Reload fulfillment URL for delivery
12. Provide product resource

**Taler (Payment)**
- Payer (Shopper) Browser
- Payee (Merchant) Site
- Taler Exchange
- Tor/HTTPS
- HTTP/HTTPS
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 $f'_{new} := f_{new} b_{new}^e \mod n$

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

Problem: Owner of $c_{new}$ may differ from owner of $c_{old}$!
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 $f'_{new} := f_{new} b_{new}^e$
Cut-and-Choose

$t_1 \rightarrow c_{old} \rightarrow \text{Lock} \rightarrow \text{Lock} \rightarrow b_{new,1}\rightarrow \text{Envelope} \rightarrow \text{transmit} \rightarrow \text{Exchange}$

$t_2 \rightarrow c_{old} \rightarrow \text{Lock} \rightarrow \text{Lock} \rightarrow b_{new,2}\rightarrow \text{Envelope} \rightarrow \text{transmit} \rightarrow \text{Exchange}$

$t_3 \rightarrow c_{old} \rightarrow \text{Lock} \rightarrow \text{Lock} \rightarrow b_{new,3}\rightarrow \text{Envelope} \rightarrow \text{transmit} \rightarrow \text{Exchange}$
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 $f'_{new, \gamma}$.
2. Compute $s' := f'_{new, \gamma}^{\frac{1}{d}} \mod n$.
3. Send signature $s'$.
Customer: Unblind change (RSA)

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

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

1. Have $c_{\text{old}}$.
2. Obtain $T_\gamma$, $s$ from exchange
3. Compute $X_\gamma = c_{\text{old}} T_\gamma$
4. Derive $c_{\text{new},\gamma}$ and $b_{\text{new},\gamma}$ from $X_\gamma$
5. Unblind $s := s' b_{\text{new},\gamma}^{-1} \mod n$
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.
- To handle protocol aborts.
Performance: taler-exchange-benchmark

Setup:
- AMD 1950X CPU
- Debian GNU/Linux
- Postgres 10.4
- Compiled with \(-O0\) – except for libgcrypt
- 800 parallel “clients” (on loopback)
- 60 reserves per client
- 15 coins per reserve
- RSA-2048
- No network latency
- No auditor

Results:
- 30% CPU Taler exchange
- 60% CPU Taler “clients”
- 3% CPU Postgres database
- \(\approx 4\, \text{ms} / \text{coin} \) (withdraw, deposit, 10% refresh chance)
  \(\Rightarrow \approx 250\, \text{transactions/s}\)

Caveats:
- 2/3rds for clients
- HTTP Keep-Alive disabled (for load-balancing)
- Used HTTP, not HTTPS
- No outgoing wire transfers
## Competitor comparison

<table>
<thead>
<tr>
<th></th>
<th>Cash</th>
<th>Bitcoin</th>
<th>Zerocoin</th>
<th>Creditcard</th>
<th>GNU Taler</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Online</strong></td>
<td>−−−−</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+++</td>
</tr>
<tr>
<td><strong>Offline</strong></td>
<td>++++</td>
<td>−−</td>
<td>−−</td>
<td>+</td>
<td>−−</td>
</tr>
<tr>
<td><strong>Trans. cost</strong></td>
<td>+</td>
<td>−−−−</td>
<td>−−−−</td>
<td>−</td>
<td>++</td>
</tr>
<tr>
<td><strong>Speed</strong></td>
<td>+</td>
<td>−−−−</td>
<td>−−−−</td>
<td>o</td>
<td>++</td>
</tr>
<tr>
<td><strong>Taxation</strong></td>
<td>−</td>
<td>−−</td>
<td>−−−−</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td><strong>Payer-anon</strong></td>
<td>++</td>
<td>o</td>
<td>++</td>
<td>−−−</td>
<td>+++</td>
</tr>
<tr>
<td><strong>Payee-anon</strong></td>
<td>++</td>
<td>o</td>
<td>++</td>
<td>−−−</td>
<td>−−−−</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>−</td>
<td>o</td>
<td>o</td>
<td>−−</td>
<td>++</td>
</tr>
<tr>
<td><strong>Conversion</strong></td>
<td>++++</td>
<td>−−−−</td>
<td>−−−−</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td><strong>Libre</strong></td>
<td>−</td>
<td>+++</td>
<td>+++</td>
<td>−−−</td>
<td>+++</td>
</tr>
</tbody>
</table>
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 SEPA, ACH, Bitcoin) are registered with IANA and allow extra options
- Under standardization with IETF as draft-dold-payto

Please voice your support!
How to support?

- Join: taler@gnu.org, #taler
- Coding & design: https://gnunet.org/bugs/
- Translation: https://git.taler.net/www.git/tree/locale/fr/LC_MESSAGES/messages.po
- Integration: https://docs.taler.net/
- Donations: https://gnunet.org/ev
- Funding: https://taler.net/en/investors.html

And of course we are looking for banks as partners!
Leon Schumacher  
co-founder

Dr. Christian Grothoff  
co-founder

Michael Widmer  
Jurist

Dr. Jeff Burdges  
PostDoc

Florian Dold  
PhD Student

Prof. Mikhail Atallah  
Cryptographer, co-founder Arxan Technologies Inc.

Prof. Roberto Di Cosmo  
Director IRILL

Greg Framke  
CIO Manulife, former COO Etrade

Ante Gulam  
Global Head of Information Security — CISO MetaPack Group

Dr. Richard Stallman  
Founder of the Free Software movement

Chris Pagett  
former Group Head Security/Fraud/Geo Risk HSBC

Prof. Alex Pentland  
MIT Media Lab
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, and
▶ Enjoy the “benefits” of cash

OR

▶ Establish free software alternative balancing social goals!
Do you have any questions?

References:


Let money facilitate trade; but ensure capital serves society.