GNU TALER

Funded by the European Union

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

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

taler.net

taler@twitter

taler-systems.com

Christian Grothoff

grothoff@taler.net

Agenda

Motivation & Background

GNU Taler: Introduction

Protocol Basics

Programmable money: Age restrictions

Blockchain integration: Project Depolymerization

Future Work & Conclusion

A Social Problem

This was a question posed to RAND researchers in 1971:

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

A Social Problem

This was a question posed to RAND researchers in 1971:

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

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

The Bank of International Settlements

GNU Taler: Introduction

Digital cash, made **socially responsible**.

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

What is Taler?

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

Taler is

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

However, Taler is

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

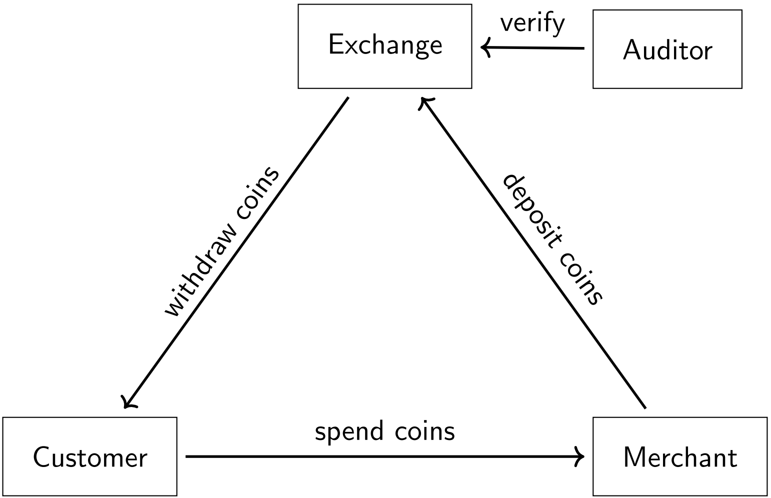
Design principles

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

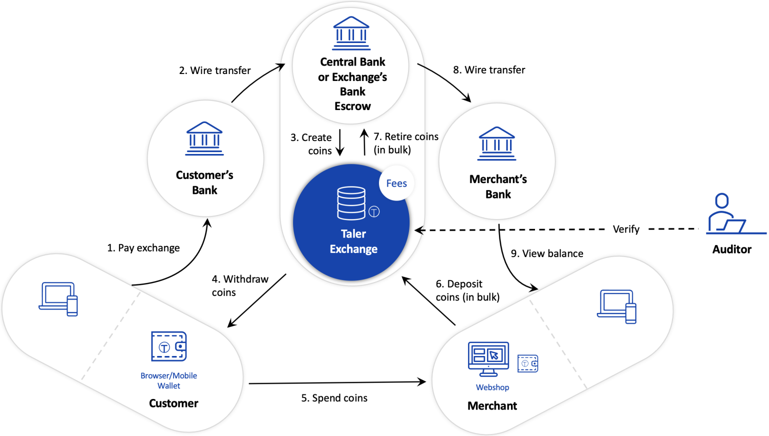
GNU Taler must ...

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



Architecture of Taler



Usability of Taler

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

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

Protocol Basics

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:

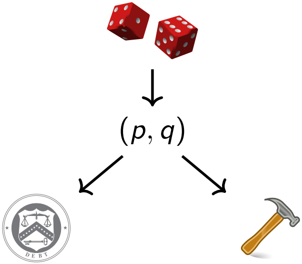
- Merchant's income is visible from deposits.
- Hash of contract is part of deposit data.
- State can trace income and enforce taxation.

Limitations:

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

Exchange setup: Create a denomination key (RSA)

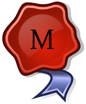
1. Pick random primes p, q.
2. Compute $n := pq$,
   $\phi(n) = (p-1)(q-1)$
3. Pick small $e < \phi(n)$ such that
   $d := e^{-1} \mod \phi(n)$ exists.
4. Publish public key (e, n).



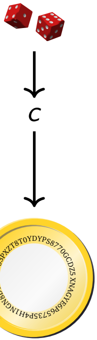
Merchant: Create a signing key (EdDSA)

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

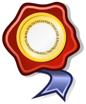
$$\downarrow$$
$$m$$
$$\downarrow$$
$$\mathrm{M}$$

Capability: $m \Rightarrow$ 

Customer: Create a planchet (EdDSA)

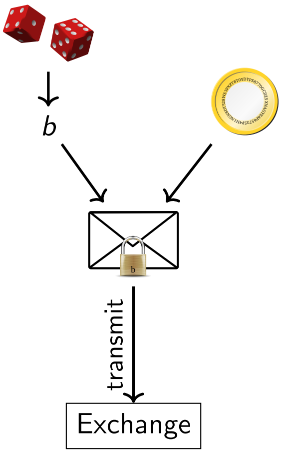


- Pick random $c \mod o$ private key
- $C = cG$ public key

Capability: $c \Rightarrow$ 

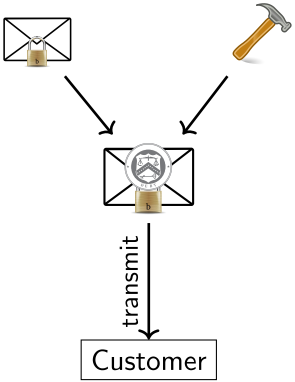
Customer: Blind planchet (RSA)

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



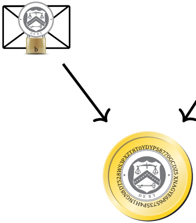
Exchange: Blind sign (RSA)

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

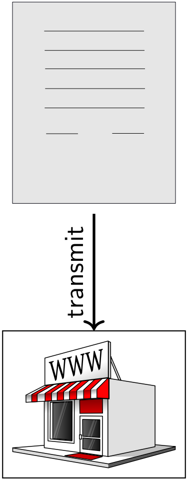


Customer: Unblind coin (RSA)

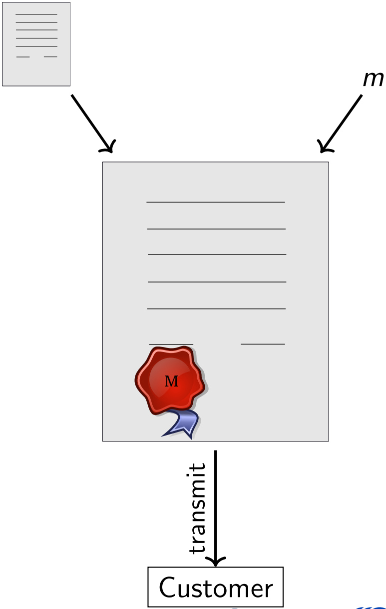
1. Receive s'.
2. Compute $s := s' b^{-1} \mod n$



Customer: Build shopping cart



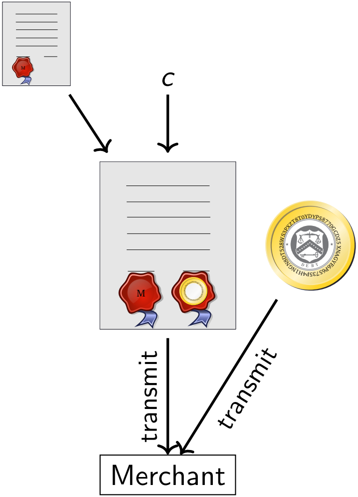
Merchant: Propose contract (EdDSA)



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

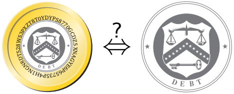
Customer: Spend coin (EdDSA)

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



Merchant 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.

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:

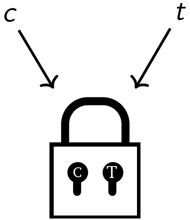
- maintain unlinkability
- maintain taxability of transactions

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
   $cT = c(tG) = t(cG) = tC$

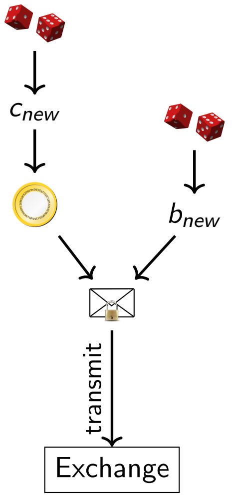


Strawman solution

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}.

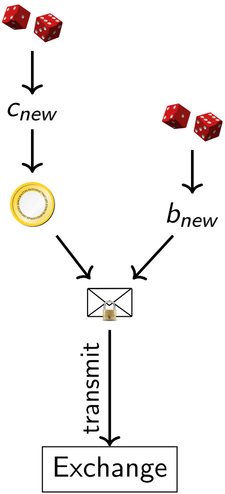


Strawman solution

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}.

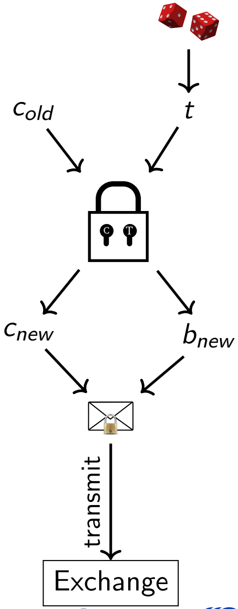


Problem: Owner of c_{new} may differ from owner of c_{old}!

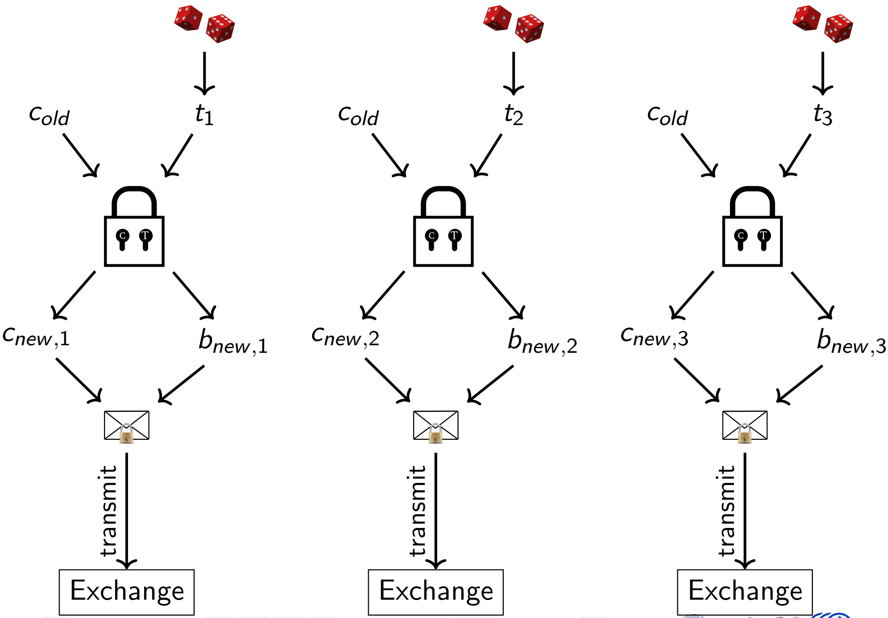
Customer: Transfer key setup (ECDH)

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.

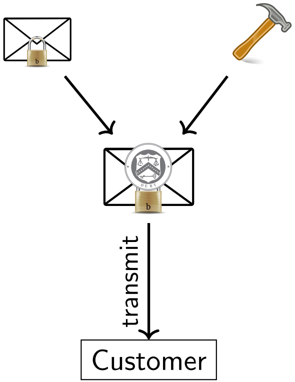
Customer: Reveal

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 ($\gamma = 2$)

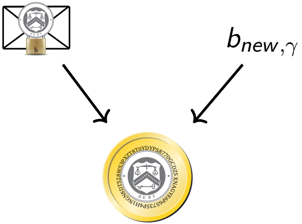
Exchange: Blind sign change (RSA)

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



Customer: Unblind change (RSA)

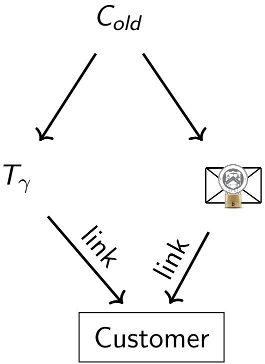
1. Receive s'.
2. Compute $s := s' b_{new,\gamma}^{-1} \mod n$.



Exchange: Allow linking change

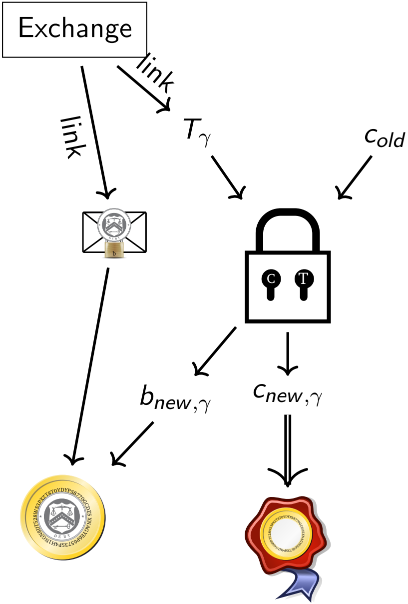
Given C_{old}

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.

Programmable money: Age restrictions

Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

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

Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

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

Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

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

Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

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

Principle of Subsidiarity is violated

Principle of Subsidiarity

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

Principle of Subsidiarity

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

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

Parents, guardians and caretakers

Age restriction design for GNU Taler

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

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

Age restriction

Assumptions and scenario

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

Age restriction

Assumptions and scenario

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

Age restriction

Assumptions and scenario

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

Age restriction

Assumptions and scenario

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

Age restriction

Assumptions and scenario

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

Age restriction

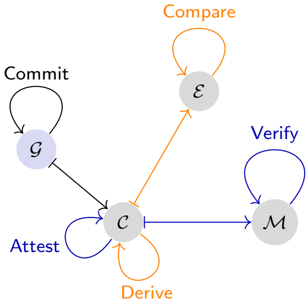
Assumptions and scenario

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

Age restriction

Assumptions and scenario

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



Note: Scheme is independent of payment service protocol.

Formal Function Signatures

Searching for functions

- Commit
- Attest
- Verify
- Derive
- Compare

Formal Function Signatures

Searching for functions with the following signatures

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

Attest

Verify

Derive

Compare

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

Formal Function Signatures

Searching for functions with the following signatures

$$\begin{array}{lll}
\mathsf{Commit}: & (\mathsf{a}, \omega) \mapsto (\mathsf{Q}, \mathsf{P}) & \mathbb{N}_\mathsf{M} \times \Omega \to \mathbb{O} \times \mathbb{P}, \\
\mathsf{Attest}: & (\mathsf{m}, \mathsf{Q}, \mathsf{P}) \mapsto \mathsf{T} & \mathbb{N}_\mathsf{M} \times \mathbb{O} \times \mathbb{P} \to \mathbb{T} \cup \{\bot\}, \\
\mathsf{Verify} & & \\
\mathsf{Derive} & & \\
\mathsf{Compare} & &
\end{array}$$

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

Formal Function Signatures

Searching for functions with the following signatures

$$\begin{aligned}
\text{Commit}: &\quad (a, \omega) \mapsto (Q, P) && \mathbb{N}_M \times \Omega \to \mathbb{O} \times \mathbb{P}, \\
\text{Attest}: &\quad (m, Q, P) \mapsto T && \mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \to \mathbb{T} \cup \{\bot\}, \\
\text{Verify}: &\quad (m, Q, T) \mapsto b && \mathbb{N}_M \times \mathbb{O} \times \mathbb{T} \to \mathbb{Z}_2, \\
\text{Derive} & \\
\text{Compare} &
\end{aligned}$$

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

Formal Function Signatures

Searching for functions with the following signatures

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

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

Formal Function Signatures

Searching for functions with the following signatures

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

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

Formal Function Signatures

Searching for functions with the following signatures

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

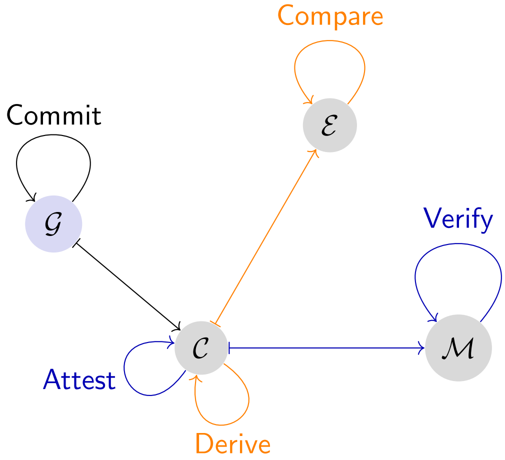
with $\Omega, \mathbb{P}, \mathbb{O}, \mathbb{T}, \mathbb{B}$ sufficiently large sets.

Basic and security requirements are defined later.

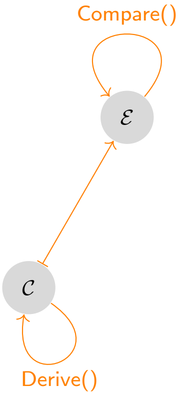
Mnemonics:
$\mathbb{O}$ = *c$\mathbb{O}$mmitments*, Q = *Q-mitment* (commitment), $\mathbb{P}$ = *$\mathbb{P}$roofs*, P = P*roof*,
$\mathbb{T}$ = *a$\mathbb{T}$testations*, T = *a*T*testation*, $\mathbb{B}$ = *$\mathbb{B}$lindings*, β = *βlinding*.

Age restriction

Naïve scheme

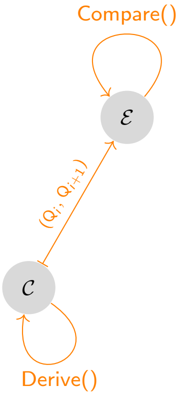


Achieving Unlinkability



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

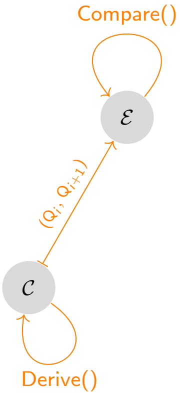
Achieving Unlinkability



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

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

Achieving Unlinkability



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

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

$\implies$ **Exchange identifies sequence**

$\implies$ **Unlinkability broken**

Achieving Unlinkability

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

Achieving Unlinkability

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

Sketch:

1. $\mathcal{C}$ derives commitments $(\mathsf{Q}_1, \dots, \mathsf{Q}_\kappa)$ from Q_0 by calling Derive() with blindings $(\beta_1, \dots, \beta_\kappa)$
2. $\mathcal{C}$ calculates $h_0 := H\left(H(\mathsf{Q}_1, \beta_1) || \dots || H(\mathsf{Q}_\kappa, \beta_\kappa)\right)$
3. $\mathcal{C}$ sends Q_0 and h_0 to $\mathcal{E}$
4. $\mathcal{E}$ chooses $\gamma \in \{1, \dots, \kappa\}$ randomly
5. $\mathcal{C}$ reveals $h_\gamma := H(\mathsf{Q}_\gamma, \beta_\gamma)$ and all (Q_i, β_i), except $(\mathsf{Q}_\gamma, \beta_\gamma)$
6. $\mathcal{E}$ compares h_0 and $H\left(H(\mathsf{Q}_1, \beta_1) || ... || h_\gamma || ... || H(\mathsf{Q}_\kappa, \beta_\kappa)\right)$ and evaluates Compare$(\mathsf{Q}_0, \mathsf{Q}_i, \beta_i)$.

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

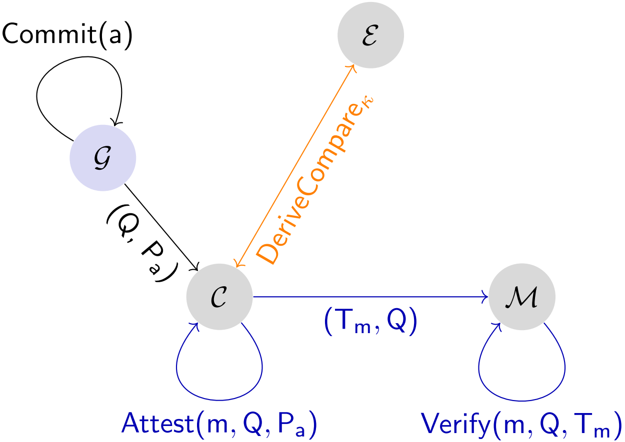
Achieving Unlinkability

With DeriveCompare$_\kappa$

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

Note: Still need Derive and Compare to be defined.

Refined scheme



Basic Requirements

Candidate functions

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

must first meet *basic* requirements:

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

Basic Requirements

Formal Details

Existence of attestations

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

Efficacy of attestations

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

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

etc.

Security Requirements

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

- Game: Age disclosure by commitment or attestation

$\leftrightarrow$ Requirement: Non-disclosure of age

- Game: Forging attestation

$\leftrightarrow$ Requirement: Unforgeability of minimum age

- Game: Distinguishing derived commitments and attestations

$\leftrightarrow$ Requirement: Unlinkability of commitments and attestations

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

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

Security Requirements

Simplified Example

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

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

Requirement: Unforgeability of minimum age

$$\underset{\mathcal{A} \in \mathfrak{A}(\mathbb{N}_{\mathsf{M}} \times \mathbb{O} \times \mathbb{P} \to \mathbb{N}_{\mathsf{M}} \times \mathbb{T})}{\forall} : \Pr\Big[G^{\mathsf{FA}}_{\mathcal{A}}(\lambda) = 1\Big] \leq \epsilon(\lambda)$$

Solution: Instantiation with ECDSA

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

Solution: Instantiation with ECDSA

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

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

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

Solution: Instantiation with ECDSA

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

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

$$\langle (q_1, p_1), \dots, (q_\mathsf{M}, p_\mathsf{M}) \rangle$$

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

$$\Big\langle (q_1, p_1), \dots, (q_\mathsf{a}, p_\mathsf{a}), (q_{\mathsf{a}+1}, \perp), \dots, (q_\mathsf{M}, \perp) \Big\rangle$$

- $\vec{\mathsf{Q}} := (q_1, \dots, q_\mathsf{M})$ is the *Commitment*,
- $\vec{\mathsf{P}}_\mathsf{a} := (p_1, \dots, p_\mathsf{a}, \perp, \dots, \perp)$ is the *Proof*

Solution: Instantiation with ECDSA

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

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

$$\langle (q_1, p_1), \ldots, (q_\mathrm{M}, p_\mathrm{M}) \rangle$$

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

$$\Big\langle (q_1, p_1), \ldots, (q_\mathrm{a}, p_\mathrm{a}), (q_{\mathrm{a}+1}, \bot), \ldots, (q_\mathrm{M}, \bot) \Big\rangle$$

   - $\vec{\mathrm{Q}} := (q_1, \ldots, q_\mathrm{M})$ is the *Commitment*,
   - $\vec{\mathrm{P}}_\mathrm{a} := (p_1, \ldots, p_\mathrm{a}, \bot, \ldots, \bot)$ is the *Proof*

3. Guardian gives child $\langle \vec{\mathrm{Q}}, \vec{\mathrm{P}}_\mathrm{a} \rangle$

Instantiation with ECDSA

Definitions of Attest and Verify

Child has

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

Instantiation with ECDSA

Definitions of Attest and Verify

Child has

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

To Attest a minimum age $\mathsf{m} \leq \mathsf{a}$:

Sign a message with ECDSA using private key p_m

Instantiation with ECDSA

Definitions of Attest and Verify

Child has

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

To Attest a minimum age $\mathsf{m} \leq \mathsf{a}$:

Sign a message with ECDSA using private key p_m

Merchant gets

- ordered public-keys $\vec{\mathsf{Q}} = (q_1, \dots, q_\mathsf{M})$
- Signature σ

Instantiation with ECDSA

Definitions of Attest and Verify

Child has

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

To **Attest** a minimum age $\mathsf{m} \leq \mathsf{a}$:

Sign a message with ECDSA using private key p_m

Merchant gets

- ordered public-keys $\vec{\mathsf{Q}} = (q_1, \dots, q_\mathsf{M})$
- Signature σ

To **Verify** a minimum age m:

Verify the ECDSA-Signature σ with public key q_m.

Instantiation with ECDSA

Definitions of Derive and Compare

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

Instantiation with ECDSA

Definitions of Derive and Compare

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

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

$$\vec{Q}' := \left(\beta * q_1, \ldots, \beta * q_M\right),$$
$$\vec{P}' := \left(\beta p_1, \ldots, \beta p_a, \perp, \ldots, \perp\right)$$

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

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

Instantiation with ECDSA

Definitions of Derive and Compare

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

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

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

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

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

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

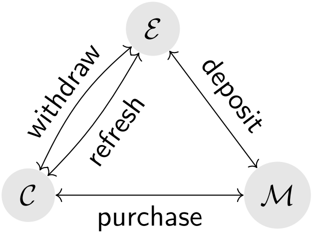
To Compare, calculate: $(\beta * q_1, \ldots, \beta * q_\mathsf{M}) \stackrel{?}{=} (q'_1, \ldots, q'_\mathsf{M})$

Instantiation with ECDSA

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

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

Reminder: GNU Taler Fundamentals



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

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

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

Integration with GNU Taler

Binding age restriction to coins

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

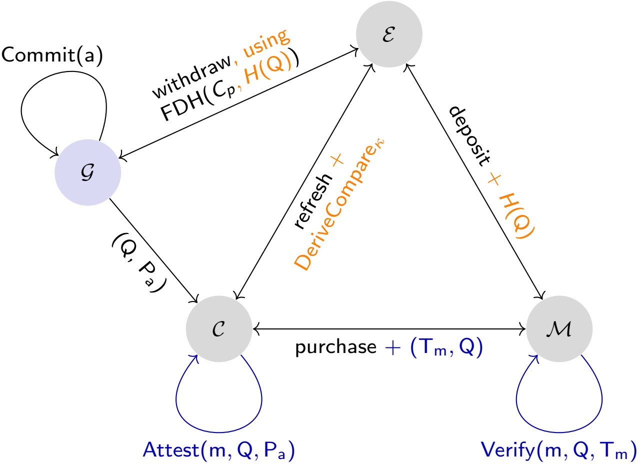
$$\text{FDH}(C_p, H(\text{Q}))$$

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

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

Integration with GNU Taler

Integrated schemes



Instantiation with Edx25519

Paper also formally defines another signature scheme: Edx25519.

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

Current implementation of age restriction in GNU Taler uses Edx25519.

Discussion

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

Related Work

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

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

Conclusion

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

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

Our scheme offers a solution that is

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

Blockchain integration: Project Depolymerization

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**

BBC | Home | News | Sport | More
NEWS | Menu
World | Africa | Asia | Australia | Europe | Latin America | Middle East | US & Canada

Kosovo bans cryptocurrency mining after blackouts

5 January

Environment ▶ Climate crisis Wildlife Energy **Pollution**

The Observer

How do we solve bitcoin's carbon problem?

The cryptocurrency consumes more energy than Norway. As countries consider copying China's ban, experts disagree on whether a greener version is possible

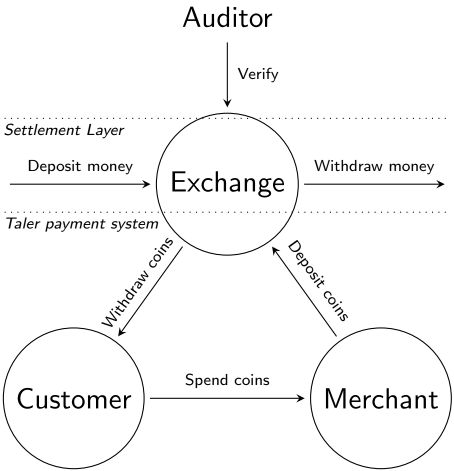
WIRED SUBSCRIBE

As Kazakhstan Descends Into Chaos, Crypto Miners Are at a Loss

The central Asian country became No. 2 in the world for Bitcoin mining. But political turmoil and power cuts have hit hard, and the future looks bleak.

Taler

Architecture



Settlement layer

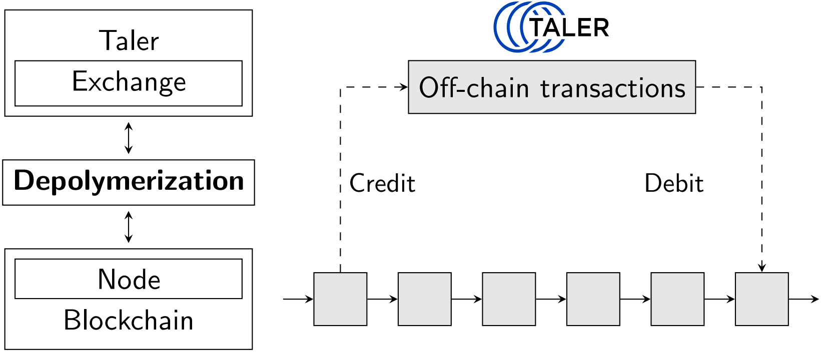
- For Depolymerization: Blockchain!

Taler payment system

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

Taler

Blockchain settlement layer



Challenges

Taler Metadata

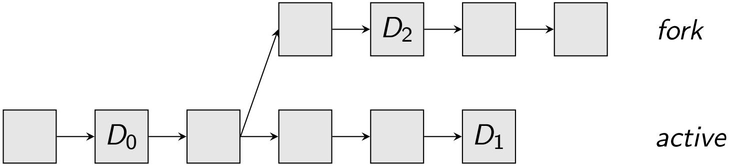
- Metadata are required to link a wallet to credits and allow merchant to link deposits to debits
- Putting metadata in blockchain transactions can be tricky

Blockchain based cryptocurrencies

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

Blockchain challenges

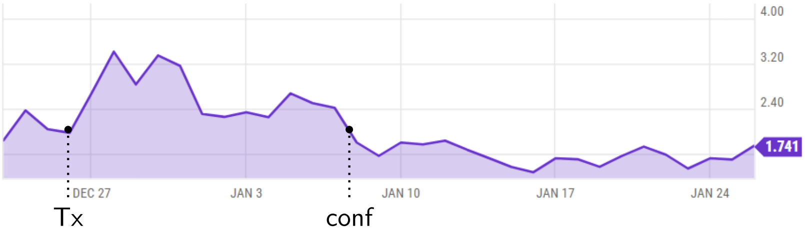
Chain reorganization



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.

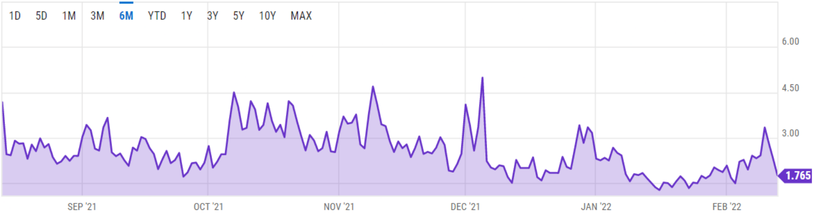
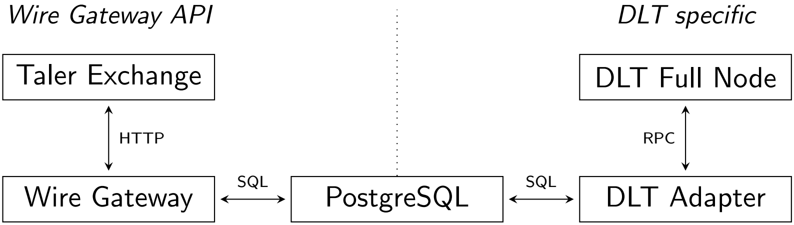


Figure: Bitcoin average transaction fee over 6 months (ychart)

However, transaction fees are unpredictable.

Depolymerization

Architecture



- 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

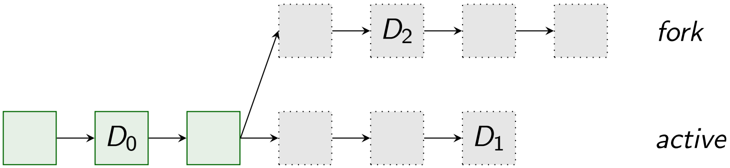
Smart contract ?

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

Custom input format

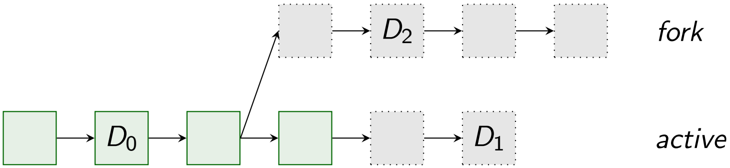
Use input data in transactions, usually used to call smart contract, to store our metadata.

Handling blockchain reorganization



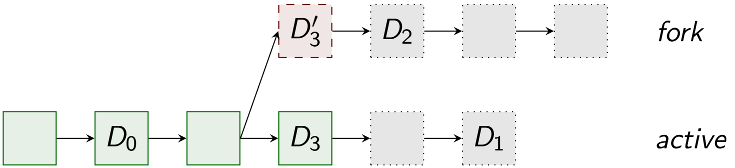
As small reorganizations are common, Satoshi already recommended to apply a confirmation delay to handle most disturbances and attacks.

Handling blockchain reorganization



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

Handling blockchain reorganization



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

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

Future Work & Conclusion

How to support?

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

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

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

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

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

Donate: https://gnunet.org/ev

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

Do you have any questions?

References:

1. Özgür Kesim, Christian Grothoff, Florian Dold and Martin Schanzenbach. *Zero-Knowledge Age Restriction for GNU Taler*. **27th European Symposium on Research in Computer Security (ESORICS), 2022**.
2. David Chaum, Christian Grothoff and Thomas Moser. *How to issue a central bank digital currency*. **SNB Working Papers, 2021**.
3. David Chaum, Christian Grothoff and Thomas Moser. *How to issue a central bank digital currency*. **SNB Working Papers, 2021**.
4. 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.
5. Jeffrey Burdges, Florian Dold, Christian Grothoff and Marcello Stanisci. *Enabling Secure Web Payments with GNU Taler*. **SPACE 2016**.
6. Florian Dold, Sree Harsha Totakura, Benedikt Müller, Jeffrey Burdges and Christian Grothoff. *Taler: Taxable Anonymous Libre Electronic Reserves*. Available upon request. 2016.
7. 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**.
8. David Chaum, Amos Fiat and Moni Naor. *Untraceable electronic cash*. **Proceedings on Advances in Cryptology, 1990**.
9. Phillip Rogaway. *The Moral Character of Cryptographic Work*. **Asiacrypt**, 2015.