

NEXT GENERATION INTERNET

GNU Taler

Christian Grothoff

07.06.2024

Learning objectives

How should we pay?

Introduction to GNU Taler

How does cut-and-choose work?

How to prove protocols secure with cryptographic games?

What are the future plans for GNU Taler?

How should we pay?

Surveillance



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?”

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)

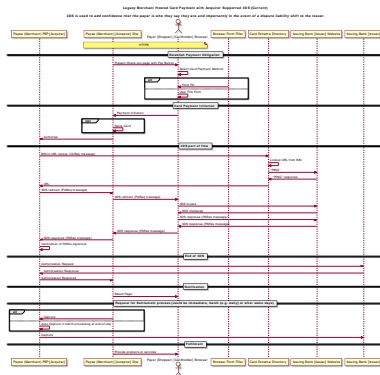
Why is it worse?

- ▶ 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 6-10 years!

Credit cards 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



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

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”

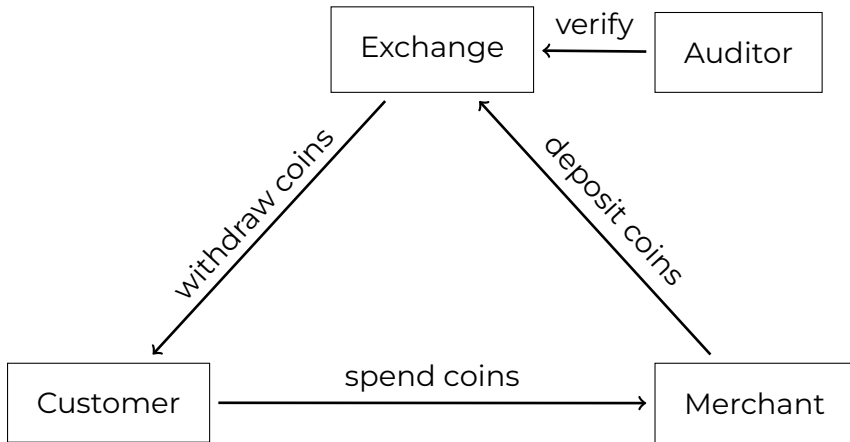
Design goals

... for the GNU Taler payment system

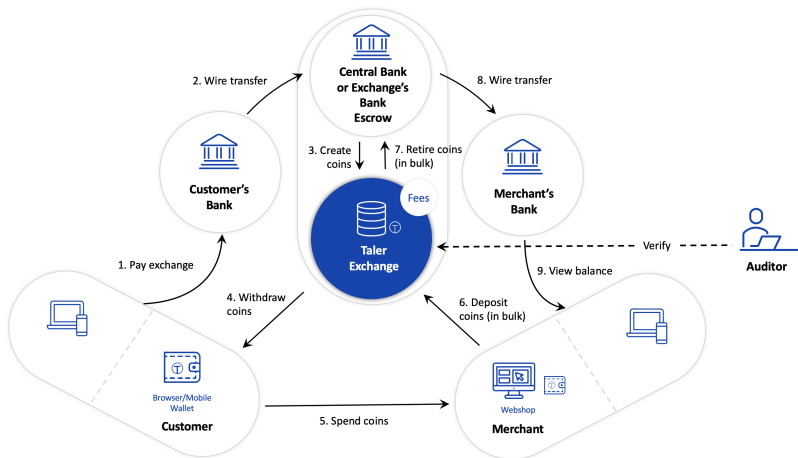
GNU Taler must ...

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



Architecture of Taler



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.

How does cut-and-choose work?

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

Other contemporary payment systems have similar limitations on identification, and thus these limitations should not be a legal issue.

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:

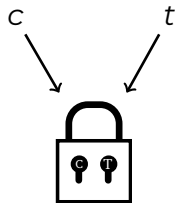
- ▶ maintain unlinkability
- ▶ maintain taxability of transactions

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)

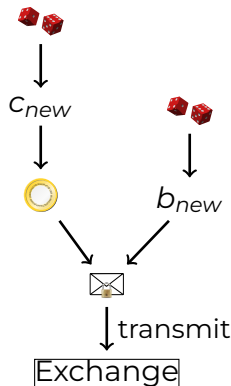
1. Create private keys $c, t \bmod o$
2. Define $C := cG$
3. Define $T := tG$
4. Compute DH:
 $cT = c(tG) = t(cG) = tC$



Strawman solution

Given partially spent private coin key c_{old} :

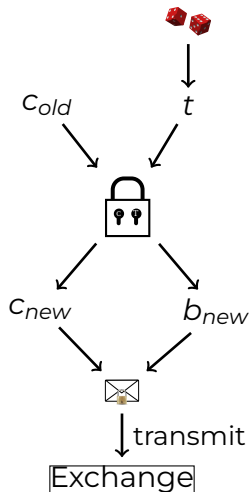
1. Pick random $c_{new} \bmod o$ private key
 2. Compute $C_{new} := c_{new}G$ public key
 3. Pick random b_{new}
 4. Compute $f_{new} := FDH(C_{new})$, $m < n$.
 5. Transmit $f'_{new} := f_{new}b_{new}^e \bmod n$
- ... and sign request for change with c_{old} .



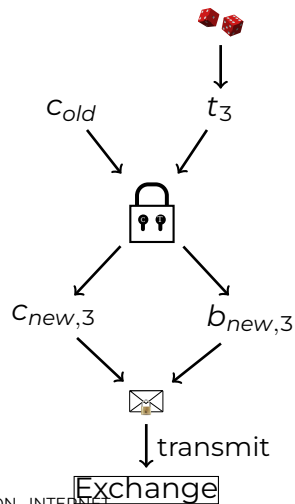
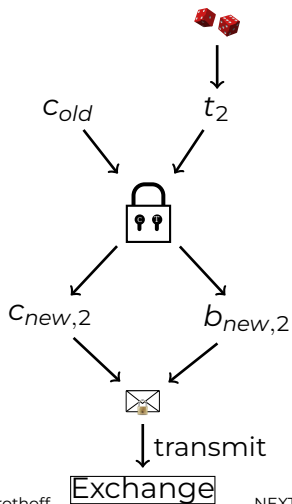
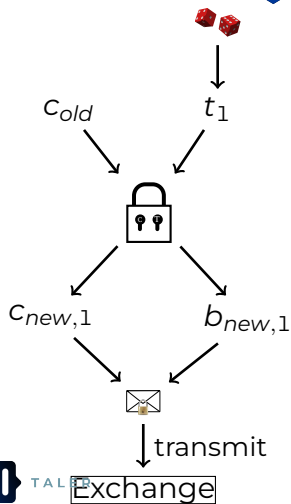
Customer: Transfer key setup (ECDH)

Given partially spent private coin key c_{old} :

1. Let $C_{old} := c_{old}G$ (as before)
2. Create random private transfer key $t \bmod o$
3. Compute public transfer key $T := tG$
4. Compute $X := c_{old}(tG) = t(c_{old}G) = tC_{old}$
5. Derive c_{new} and b_{new} from X using HKDF
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



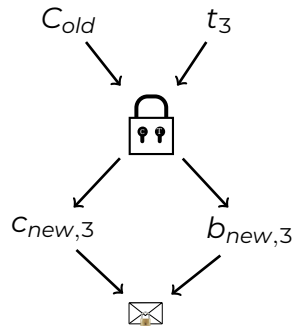
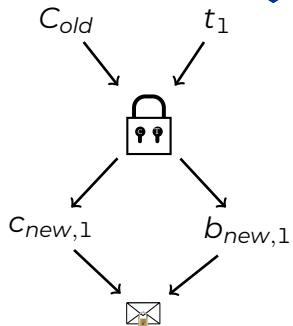
Exchange: Choose!

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

Customer: Reveal

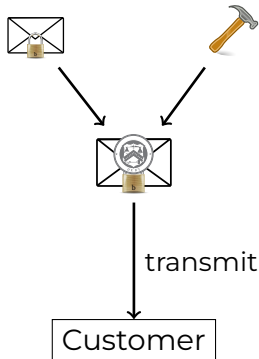
1. If $\gamma = 1$, send t_2, t_3 to exchange
2. If $\gamma = 2$, send t_1, t_3 to exchange
3. If $\gamma = 3$, send t_1, t_2 to exchange

Exchange: Verify ($\gamma = 2$)



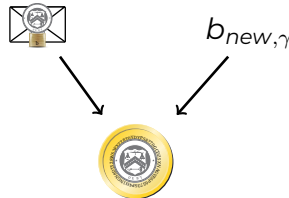
Exchange: Blind sign change (RSA)

1. Take $f'_{new,\gamma}$.
2. Compute $s' := f'^d_{new,\gamma} \bmod n$.
3. Return signature s' .



Customer: Unblind change (RSA)

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

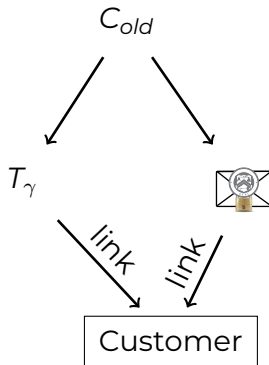


Exchange: Allow linking change

Given C_{old}

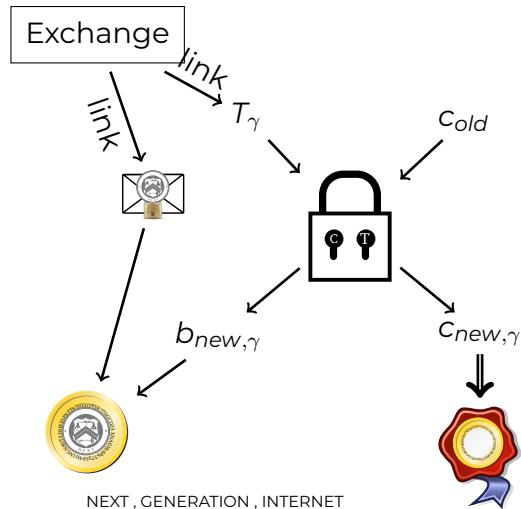
return T_γ and

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



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
- ⇒ 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.

How to prove protocols secure with cryptographic games?

Reminder: Cryptographic games

An *oracle* is a party in a game that the adversary can call upon to indirectly access information that is otherwise hidden from it.

For example, **IND-CPA** can be formalized like this:

Setup Generate random key k , select $b \in \{0, 1\}$ for $i \in \{1, \dots, q\}$.

Oracle Given M_0 and M_1 (of same length), return $C := \text{enc}(k, M_b)$.

The adversary wins, if it can guess b with probability greater than $\frac{1}{2} + \epsilon(\kappa)$ where $\epsilon(\kappa)$ is a negligible function in the security parameter κ .

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

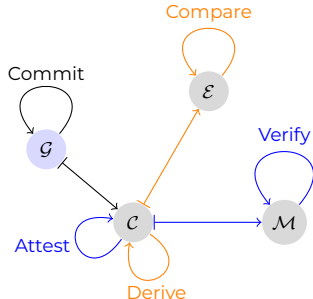
Design goals

1. Tie age restriction to the **ability to pay** (not to ID's)
2. maintain **anonymity of buyers**
3. maintain **unlinkability of transactions**
4. align with **principle of subsidiarity**
5. be **practical and efficient**

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 (minimum) age
- ▶ *Merchants* **verify** the attestations
- ▶ Minors **derive** age commitments from existing ones
- ▶ *Exchanges* **compare** the derived age commitments



Formal function signatures

Searching for functions with the following signatures

Commit :	$(a, \omega) \mapsto (Q, P)$	$N_M \times \Omega \rightarrow \mathbb{O} \times \mathbb{P},$
Attest :	$(m, Q, P) \mapsto T$	$N_M \times \mathbb{O} \times \mathbb{P} \rightarrow \mathbb{T} \cup \{\perp\},$
Verify :	$(m, Q, T) \mapsto b$	$N_M \times \mathbb{O} \times \mathbb{T} \rightarrow \mathbb{Z}_2,$
Derive :	$(Q, P, \omega) \mapsto (Q', P', \beta)$	$\mathbb{O} \times \mathbb{P} \times \Omega \rightarrow \mathbb{O} \times \mathbb{P} \times \mathbb{B},$
Compare :	$(Q, Q', \beta) \mapsto b$	$\mathbb{O} \times \mathbb{O} \times \mathbb{B} \rightarrow \mathbb{Z}_2,$

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

Basic and security requirements are defined later.

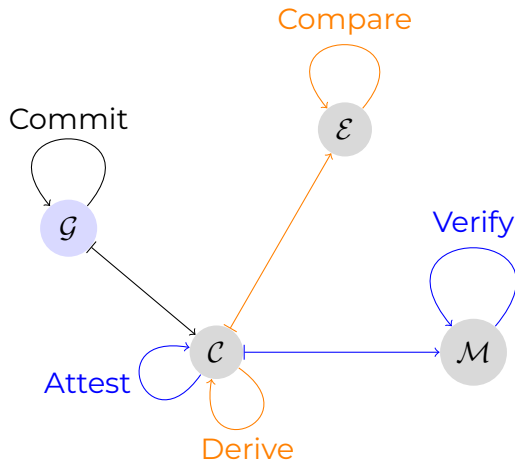
Mnemonics:

$\mathbb{O} = c\mathbb{O}mmitments, Q = Q\text{-}mitment$ (commitment), $\mathbb{P} = \mathbb{P}roofs, P = P\text{-}roof,$

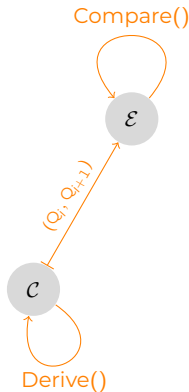
$\mathbb{T} = a\mathbb{T}testations, T = a\mathbb{T}testation, \mathbb{B} = \mathbb{B}lindings, \beta = \beta\text{-}linding.$

Age restriction

Naïve scheme



Achieving unlinkability



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

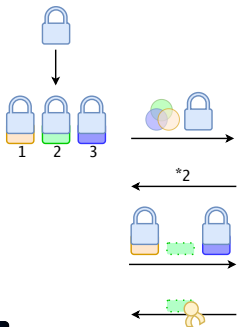
- ▶ Calling Derive() iteratively generates sequence (Q_0, Q_1, \dots) of commitments.
- ▶ Exchange calls Compare(Q_i, Q_{i+1}, \dots)

⇒ **Exchange identifies sequence**

⇒ **Unlinkability broken**

Achieving unlinkability

Define cut&choose protocol **DeriveCompare _{κ}** , using **Derive()** and **Compare()**, sketch:



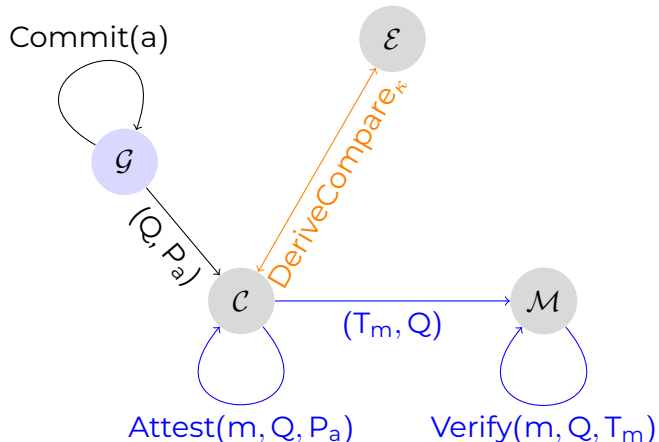
1. \mathcal{C} derives commitments (Q_1, \dots, Q_κ) from Q_0 by calling **Derive()** with blindings $(\beta_1, \dots, \beta_\kappa)$
2. \mathcal{C} calculates $h_0 := H(H(Q_1, \beta_1) || \dots || H(Q_\kappa, \beta_\kappa))$
3. \mathcal{C} sends Q_0 and h_0 to \mathcal{E}
4. \mathcal{E} chooses $\gamma \in \{1, \dots, \kappa\}$ randomly
5. \mathcal{C} reveals $h_\gamma := H(Q_\gamma, \beta_\gamma)$ and all (Q_i, β_i) , except (Q_γ, β_γ)
6. \mathcal{E} compares h_0 and $H(H(Q_1, \beta_1) || \dots || h_\gamma || \dots || H(Q_\kappa, \beta_\kappa))$ and evaluates **Compare** (Q_0, Q_i, β_i) .

Achieving unlinkability

With **DeriveCompare _{κ}**

- ▶ \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.



Basic requirements

Candidate functions

(Commit, Attest, Verify, Derive, Compare)

must first meet *basic* requirements:

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

Basic requirements

Formal details

Existence of attestations

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

Efficacy of attestations

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

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

etc.

Security requirements

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

- ▶ Game: Age disclosure by commitment or attestation
- ↔ Requirement: Non-disclosure of age
- ▶ Game: Forging attestation
- ↔ Requirement: Unforgeability of minimum age
- ▶ Game: Distinguishing derived commitments and attestations
- ↔ 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

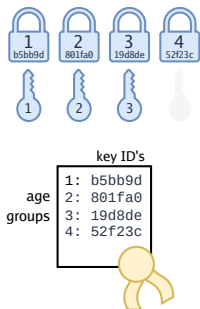
Game $G_{\mathcal{A}}^{\text{FA}}(\lambda)$ —Forging an attest:

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

Requirement: Unforgeability of minimum age

$$\forall \mathcal{A} \in \mathfrak{A}(\mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \rightarrow \mathbb{N}_M \times \mathbb{T}) : \Pr[G_{\mathcal{A}}^{\text{FA}}(\lambda) = 1] \leq \epsilon(\lambda)$$

Solution: Instantiation with ECDSA



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

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

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

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

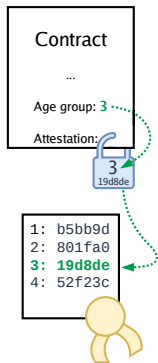
$$\langle (q_1, p_1), \dots, (q_a, p_a), (q_{a+1}, \perp), \dots, (q_M, \perp) \rangle$$

- ▶ $\vec{Q} := (q_1, \dots, q_M)$ is the *Commitment*,
- ▶ $\vec{P}_a := (p_1, \dots, p_a, \perp, \dots, \perp)$ is the *Proof*

3. Guardian gives child $\langle \vec{Q}, \vec{P}_a \rangle$

Instantiation with ECDSA

Definitions of Attest and Verify



Child has

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

To **Attest** a minimum age $m \leq a$:

Sign a message with ECDSA using private key

p_m

Merchant gets

- ▶ ordered public-keys $\vec{Q} = (q_1, \dots, q_M)$
- ▶ Signature σ

To **Verify** a minimum age m :

Verify the ECDSA-Signature σ with public key

q_m .

Definitions of Derive and Compare

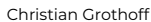
To Derive new \vec{Q}' and \vec{P}' :

$$\begin{aligned}\vec{Q}' &:= (\beta * q_1, \dots, \beta * q_M), \\ \vec{P}' &:= (\beta p_1, \dots, \beta p_a, \perp, \dots, \perp)\end{aligned}$$

Exchange gets $\vec{Q} = (q_1, \dots, q_M)$, $\vec{Q}' = (q'_1, \dots, q'_M)$
and β

To Compare, calculate:

$$(\beta * q_1, \dots, \beta * q_M) \stackrel{?}{=} (g'_1, \dots, g'_M)$$



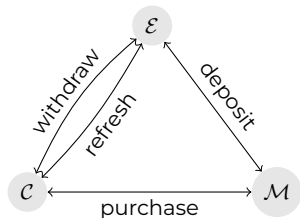
Instantiation with ECDSA

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.

Integration with GNU Taler

Key operations in the original system



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

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

(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

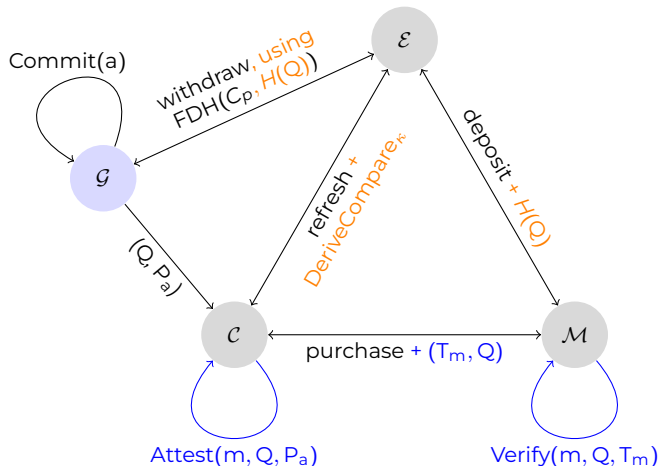
$$\text{FDH}(C_p, H(Q))$$

Verification of a coin now requires $H(Q)$, too:

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

Integration with GNU Taler

Integrated schemes



Instantiation with Edx25519

Paper also formally defines another signature scheme: Edx25519.

- ▶ Scheme already in use in GUNet,
- ▶ 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

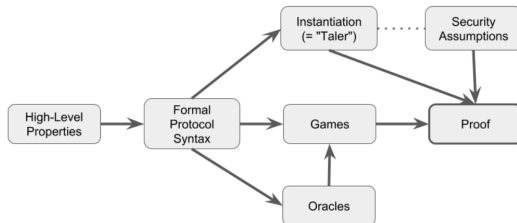
- ▶ Approach can be used with any token-based payment scheme
- ▶ Subsidiarity requires bank accounts being owned by adults
- ▶ Scheme can be adapted to case where minors have bank accounts
 - ▶ Assumption: banks provide minimum age information during bank transactions.
 - ▶ Child and Exchange execute a variant of the cut&choose protocol.

What are the future plans for GNU Taler?

Summary

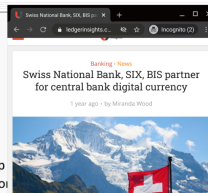
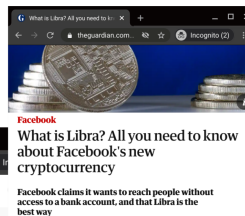
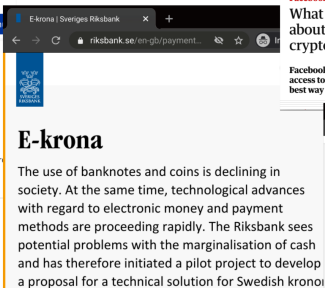
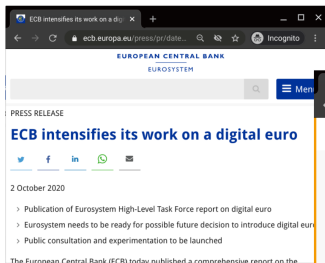
GNU Taler:

- ▶ Gives change, can provide refunds
- ▶ Integrates nicely with HTTP, handles network failures
- ▶ Has high performance
- ▶ Is Free Software
- ▶ Includes formal security proofs



CBDC initiatives and GNU Taler

Many initiatives are currently at the level of requirements discussion:



Unique regulatory features for CBs

1. Central bank issues digital coins equivalent to issuing cash
2. Architecture with consumer accounts at commercial banks
3. Withdrawal limits and denomination expiration
4. Income transparency and possibility to set fees
5. Revocation protocols and loss limitations
6. Privacy by cryptographic design not organizational compliance

Political support needed, talk to your representatives!

Ongoing work

- ▶ Post-quantum blind signatures
- ▶ Integration into more physical machines
- ▶ Integration with KYC/AML providers
- ▶ Deployment for regional currency in Basel
- ▶ Integration with Swiss Postfinance EBICS API
- ▶ Wallet backup and recovery with Anastasis
- ▶ Internationalization ⇒ <https://weblate.taler.net/>

<https://bugs.taler.net/> tracks open issues.

Open issues / Future Work

- ▶ Unlinkable subscriptions and discounts
- ▶ Privacy-preserving donations
- ▶ Integration with e-commerce frameworks (Prestashop, OpenCart, ECWID, ...)
- ▶ Currency conversion
- ▶ SAP integration
- ▶ Design and usability for illiterate and innumerate users
- ▶ Federated exchange (wads)
- ▶ ...

Help needed, talk to us (e.g. at <https://ich.taler.net/>)

Visions

- ▶ Be paid to read advertising, starting with spam
- ▶ Give welfare without intermediaries taking huge cuts
- ▶ Foster 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 I

Acknowledgements

Co-funded by the European Union (Project 101135475).



**Co-funded by
the European Union**

Co-funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by



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**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union.
Neither the European Union nor the granting authority can be held responsible for them.