Hacking in Parallel

Are you old enough to buy this?

Zero-Knowledge Age Restriction for GNU Taler

Özgür Kesim

December 29, 2022

FU Berlin

Chapters

1. TODO: something something

Introduction

Who am I

TODO: who am i

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

Our contribution

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

Age restriction

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

Age restriction

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

Age restriction

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

Age restriction

- 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

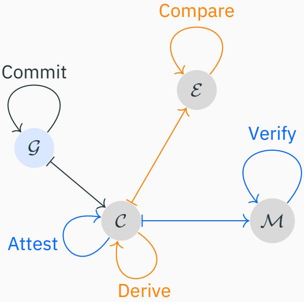
- Assumption: Checking accounts are under control of eligible adults/guardians.
- *Guardians* **commit** to an maximum age
- *Minors* **attest** their adequate age
- *Merchants* **verify** the attestations
- Minors **derive** age commitments from existing ones

Age restriction

- 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

- 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

$$
\begin{aligned}
&\mathrm{Commit}: && (\mathsf{a}, \omega) \mapsto (\mathsf{Q}, \mathsf{P}) && \mathbb{N}_{\mathsf{M}} \times \Omega \to \mathbb{O} \times \mathbb{P},\\
&\mathrm{Attest} && && \\
&\mathrm{Verify} && && \\
&\mathrm{Derive} && && \\
&\mathrm{Compare} && &&
\end{aligned}
$$

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

$$
\begin{aligned}
\mathsf{Commit}: &\quad (\mathsf{a}, \omega) \mapsto (\mathsf{Q}, \mathsf{P}) &\quad \mathbb{N}_{\mathsf{M}} \times \Omega \to \mathbb{O} \times \mathbb{P}, \\
\mathsf{Attest}: &\quad (\mathsf{m}, \mathsf{Q}, \mathsf{P}) \mapsto \mathsf{T} &\quad \mathbb{N}_{\mathsf{M}} \times \mathbb{O} \times \mathbb{P} \to \mathbb{T} \cup \{\perp\},
\end{aligned}
$$

Verify

Derive

Compare

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

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

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

Formal Function Signatures

Searching for functions with the following signatures

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

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

Formal Function Signatures

Searching for functions with the following signatures

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

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

Formal Function Signatures

Searching for functions with the following signatures

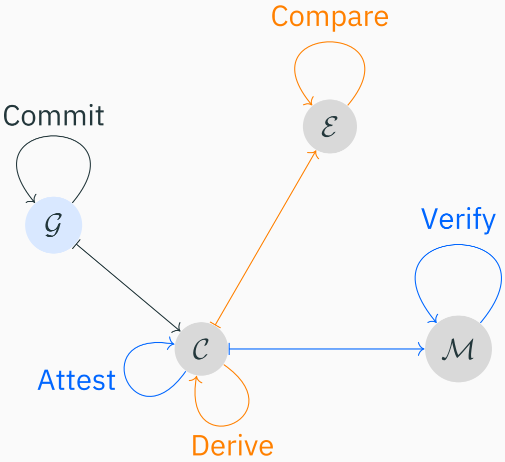
$$\begin{aligned}
\text{Commit}: &\quad (\mathsf{a}, \omega) \mapsto (\mathsf{Q}, \mathsf{P}) && \mathbb{N}_{\mathsf{M}} \times \Omega \to \mathbb{O} \times \mathbb{P}, \\
\text{Attest}: &\quad (\mathsf{m}, \mathsf{Q}, \mathsf{P}) \mapsto \mathsf{T} && \mathbb{N}_{\mathsf{M}} \times \mathbb{O} \times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

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

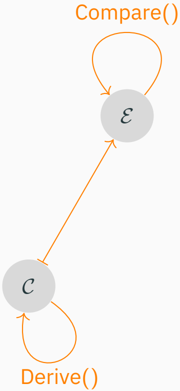
Basic and security requirements are defined later.

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

Age restriction

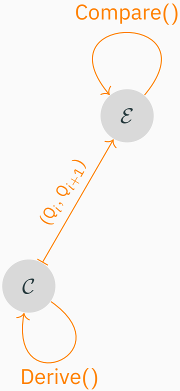


Achieving Unlinkability



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

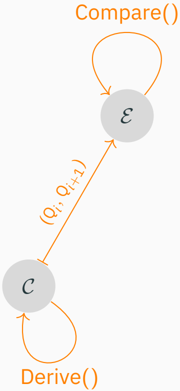
Achieving Unlinkability



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

- Calling Derive() iteratively generates sequence $(Q_0, Q_1, \dots)$ 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$(Q_i, Q_{i+1}, .)$

$\Longrightarrow$ **Exchange identifies sequence**

$\Longrightarrow$ **Unlinkability broken**

Achieving Unlinkability

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

Achieving Unlinkability

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

Sketch:

1. $\mathcal{C}$ derives commitments $(Q_1, \dots, Q_\kappa)$ from Q_0 by calling Derive() with blindings $(\beta_1, \dots, \beta_\kappa)$
2. $\mathcal{C}$ calculates $h_0 := H\left(H(Q_1, \beta_1)||\dots||H(Q_\kappa, \beta_\kappa)\right)$
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 $h_\gamma := H(Q_\gamma, \beta_\gamma)$ and all (Q_i, β_i), except (Q_γ, β_γ)
6. $\mathcal{E}$ compares h_0 and $H\left(H(Q_1, \beta_1)||...||h_\gamma||...||H(Q_\kappa, \beta_\kappa)\right)$ and evaluates $\text{Compare}(Q_0, Q_i, \beta_i)$.

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

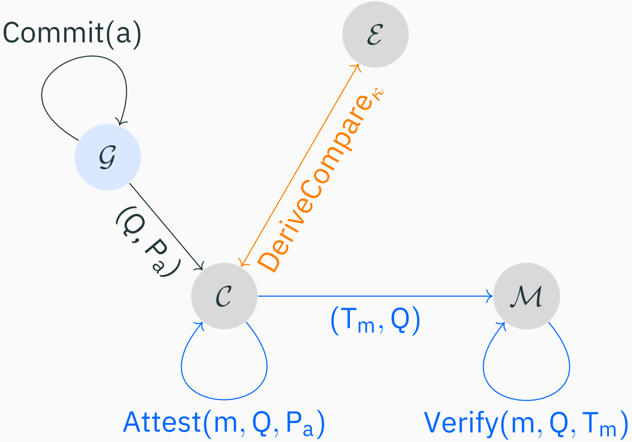
Achieving Unlinkability

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

(Commit, Attest, Verify, Derive, Compare)

must first meet *basic* requirements:

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

Basic Requirements

Existence of attestations

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

Efficacy of attestations

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

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

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

Game $G_{\mathcal{A}}^{\mathsf{FA}}(\lambda)$—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

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

Solution/Instantiation

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) $\mathsf{a} \in \{1, \ldots, \mathsf{M}\}$

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

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

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

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

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

Solution: Instantiation with ECDSA

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

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

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

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

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

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

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

Instantiation with ECDSA

Child has

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

Instantiation with ECDSA

Child has

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

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

Sign a message with ECDSA using private key p_m

Instantiation with ECDSA

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

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.

Instantiation with ECDSA

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

Instantiation with ECDSA

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

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

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

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 $\beta \in \mathbb{Z}_\mathsf{g}$ and calculate

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

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

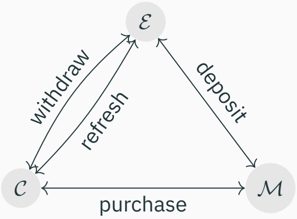
Instantiation with ECDSA

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.

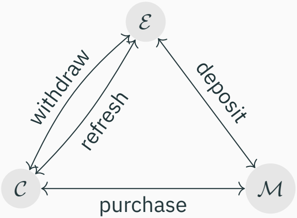
Integration with GNU Taler

GNU Taler



- Protocol suite for online payment services
- Based on Chaum's blind signatures
- Allows for change and refund (F. Dold)
- Privacy preserving: anonymous and unlinkable payments

GNU Taler



- Protocol suite for online payment services
- Based on Chaum's blind signatures
- Allows for change and refund (F. Dold)
- Privacy preserving: anonymous and unlinkable payments

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

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

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

Integration with GNU Taler

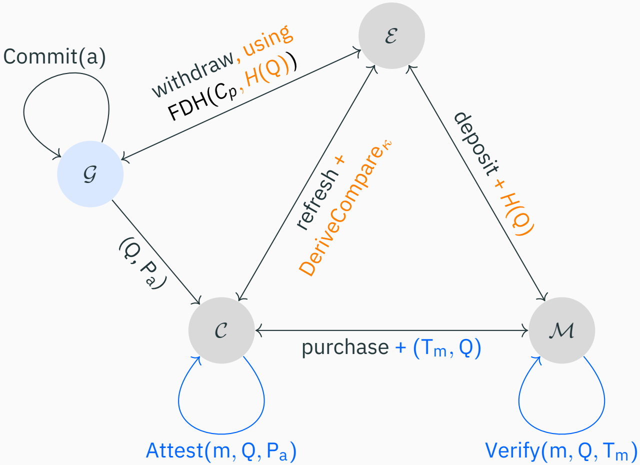
To bind an age commitment Q to a coin C_p, instead of signing FDH(C_p), $\mathcal{E}$ now blindly signs

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

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

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

Integration with GNU Taler



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 Edx2551

Discussion, Related Work, Conclusion

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

Thank you! Questions?

`oec-taler@kesim.org`

`@oec@mathstodon.xyz`

Nothing to see here