NEXT GENERATION INTERNET

The GNU Taler Payment System

Christian Grothoff

Traffic Seminar — ETHZ

Agenda

A Social Problem

This was a question posed to RAND researchers in 1971:

> *"Suppose you were an advisor to the head of the KGB. 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:

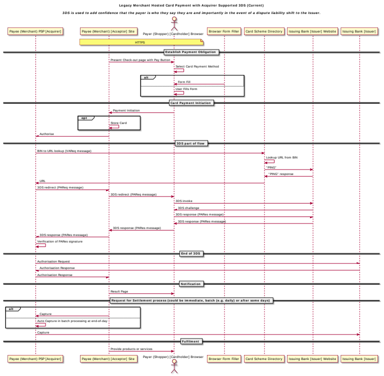
> *"Suppose you were an advisor to the head of the KGB. Suppose you are given the assignment of designing a system for the surveillance of all citizens and visitors within the boundaries of the USSR. The system is not to be too obtrusive or obvious. What would be your decision?"*

"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)

Banks 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


Legacy Merchant Hosted Card Payment with Acquirer Supported 3DS (Current)
3DS is used to add confidence that the payer is who they say they are and importantly in the event of a dispute liability shift to the Issuer.
Payee (Merchant) PSP [Acquirer]
Payee (Merchant) [Acceptor] Site
Payer (Shopper) [Cardholder] Browser
Browser Form Filler
Card Scheme Directory
Issuing Bank [Issuer] Website
Issuing Bank [Issuer]
HTTPS
Establish Payment Obligation
Present Check-out page with Pay Button
Select Card Payment Method
alt
Form Fill
User Fills Form
Card Payment Initiation
Payment Initiation
opt
Store Card
Authorise
3DS part of flow
BIN to URL lookup (VAReq message)
Lookup URL from BIN
"PING"
"PING" response
URL
3DS redirect (PAReq message)
3DS redirect (PAReq message)
3DS invoke
3DS challenge
3DS response (PARes message)
3DS response (PARes message)
3DS response (PARes message)
3DS response (PARes message)
Verification of PARes signature
End of 3DS
Authorisation Request
Authorisation Response
Authorisation Response
Notification
Result Page
Request for Settlement process (could be immediate, batch (e.g. daily) or after some days)
alt
Capture
Auto Capture in batch processing at end-of-day
Capture
Fulfilment
Provide products or services
Payee (Merchant) PSP [Acquirer]
Payee (Merchant) [Acceptor] Site
Payer (Shopper) [Cardholder] Browser
Browser Form Filler
Card Scheme Directory
Issuing Bank [Issuer] Website
Issuing Bank [Issuer]

Online credit card payments will be replaced, but with what?

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

Central Bank Digital Currency?

Speech by Augustin Carstens, Bank of International Settlements (October 2020) on the difference between Central Bank Digital Currencies and cash.

Central Bank Digital Currency vs. Cash

`https://www.youtube.com/watch?v=R_E4Uu7ycqE` (10'2020)

The Emergency Act of Canada

Speech by Premier Kenney, Alberta, February 2022.

The Emergency Act of Canada

https://www.youtube.com/watch?v=NehMAj492SA (2’2022)

GNU Taler: Introduction

GNU Taler [1, 3, 2]

Digital cash, made **socially responsible**.

TALER

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* based on proof-of-work or proof-of-stake

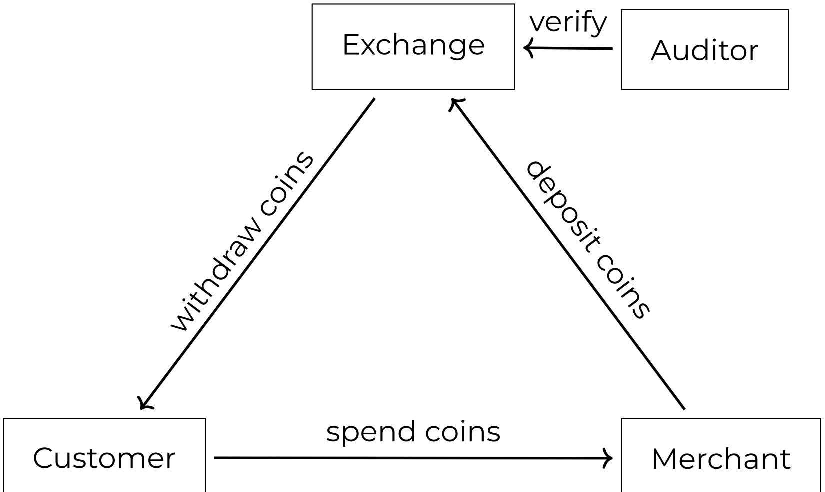
Design principles

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

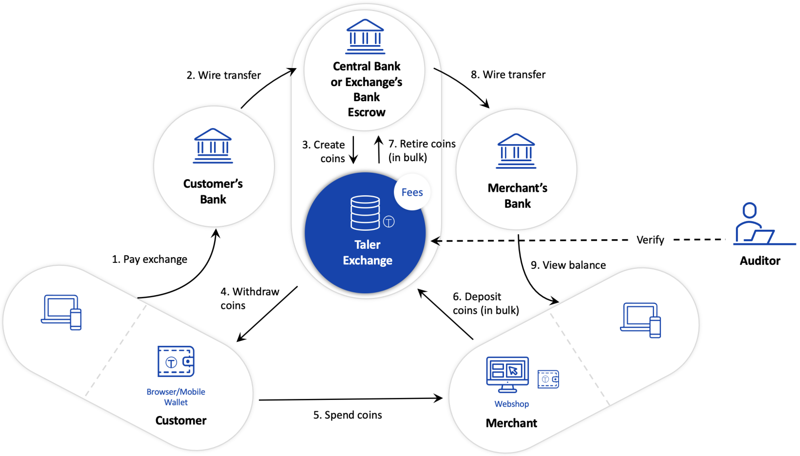
GNU Taler must ...

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



Consumer Impact of Taler

- **Convenient:** pay with one click instantly — in Euro, Dollar, Yen or Bitcoin
- **Friction-free security:** Payments do not require sign-up, login or multi-factor authentication
- **Privacy-preserving:** payment requires/shares no personal information
- **Bank account:** not required

Merchant Impact of Taler

- **Instant clearance:** one-click transactions and instant clearance at par
- **Easy & compliant:** GDPR & PCI-DSS compliance-free and without any effort
- **Major profit increase:** efficient protocol + no fraud = extremely low costs
- **1-click checkout:** without Amazon and without false positives in fraud detection

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

BFH Bachelor’s thesis video

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)~~ Deterministic signatures (1977)
- 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:

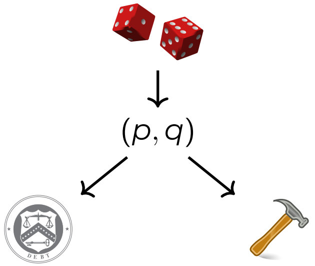
- Merchant’s income is visible from deposits.
- Hash of contract is part of deposit data.
- State can trace income and enforce taxation.

Limitations:

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

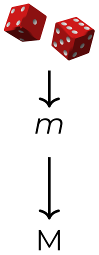
Exchange setup: Create a denomination key (RSA)

1. Generate 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).

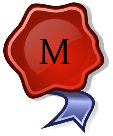


Merchant: Create a signing key (EdDSA)

- Generate random number $m \mod o$ as private key
- Compute public key $M := mG$

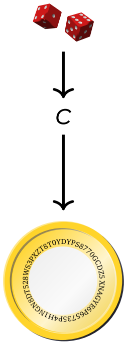


Capability:
$m \Rightarrow$



Customer: Create a planchet (EdDSA)

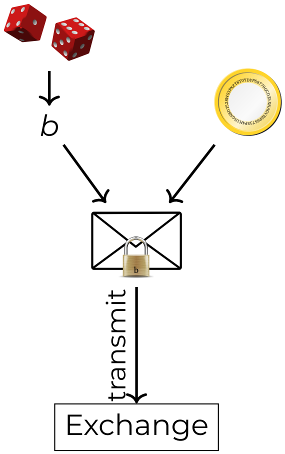
- Generate random number $c \mod o$ as private key
- Compute public key $C := cG$



Capability: $c \Rightarrow$

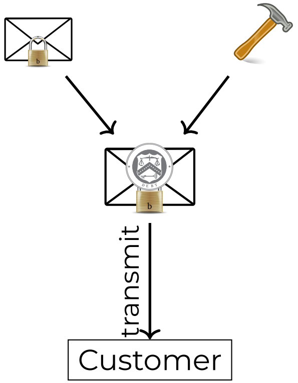
Customer: Blind planchet (RSA)

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



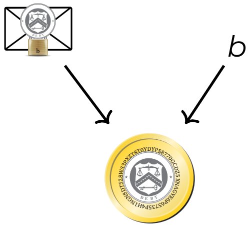
Exchange: Blind sign (RSA)

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

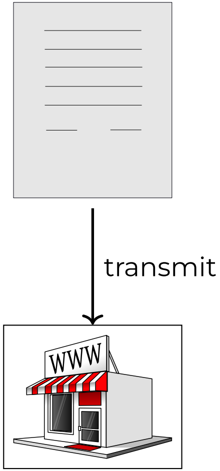


Customer: Unblind coin (RSA)

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

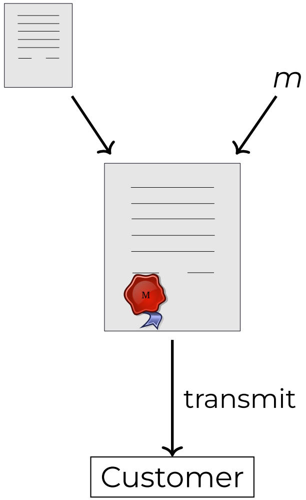


Customer: Build shopping cart



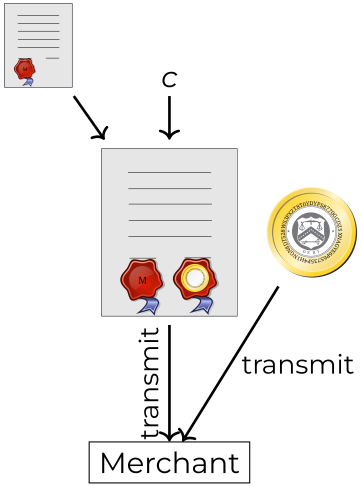
Merchant: Propose contract (EdDSA)

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



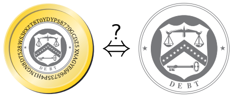
Customer: Spend coin (EdDSA)

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



Merchant and Exchange: Verify coin (RSA)

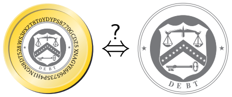
$$s^e \overset{?}{\equiv} FDH(C) \mod n$$



The exchange does not only verify the signature, but also checks that the coin was not double-spent.

Merchant and Exchange: Verify coin (RSA)

$$s^e \overset{?}{\equiv} FDH(C) \mod n$$



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.
- Wallet may not have exact change!
- Must be able 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.
- Wallet may not have exact change!
- Must be able 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.
- Wallet may not have exact change!
- Must be able to pay given sufficient total funds.

Key goals:

- maintain unlinkability
- maintain taxability of transactions

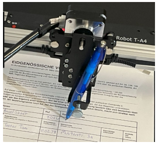
Method:

- Contract can specify to pay *partial value* of a coin.
- Allow wallet to obtain *unlinkable change*.

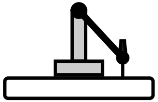
Unique Signatures

- Some public key operations depend on a nonce or “random” value
  - Ex.: DSA/ECDSA (signing)
  - + same plaintext, different ciphertext
  - - security may break on nonce-reuse
- Generating the nonce deterministically by hashing all inputs (see also: Fiat-Shamir transformation) can make these algorithms **deterministic**
  - Ex.: EdDSA
- If only one form of a valid signature exists and the verifier can check this, a signature is **unique**.
  - Ex.: RSA, Verifiable Random Func.

Unique signatures:



$=$



Verifiable Random Functions

Micali, Rabin, & Vadhan (1999) proposed verifiable random functions.

Let M be some input.

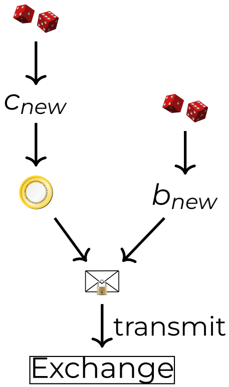
- $(sk, pk) := VRF_{keygen}()$
- $Verifier$ picks M
- $(v, p) := VRF_{sign}(M, sk)$
- v is deterministic, unpredictable and high-entropy for any M and sk, and (v, p) can only be computed with sk
- $VRF_{verify}(M, pk, v, p)$ returns true only if v was computed correctly
- sk cannot be derived from M, pk, v and p

Straw-man solution

Given partially spent private coin key c_{old}:

1. Pick random $c_{new} \mod o$ 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 $f'_{new} := f_{new}b^e_{new} \mod n$

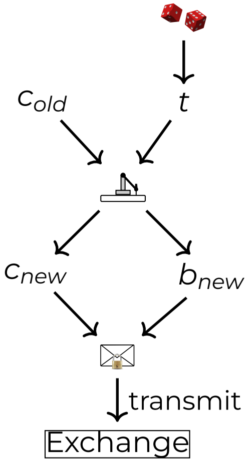
... and sign request for change with c_{old}.



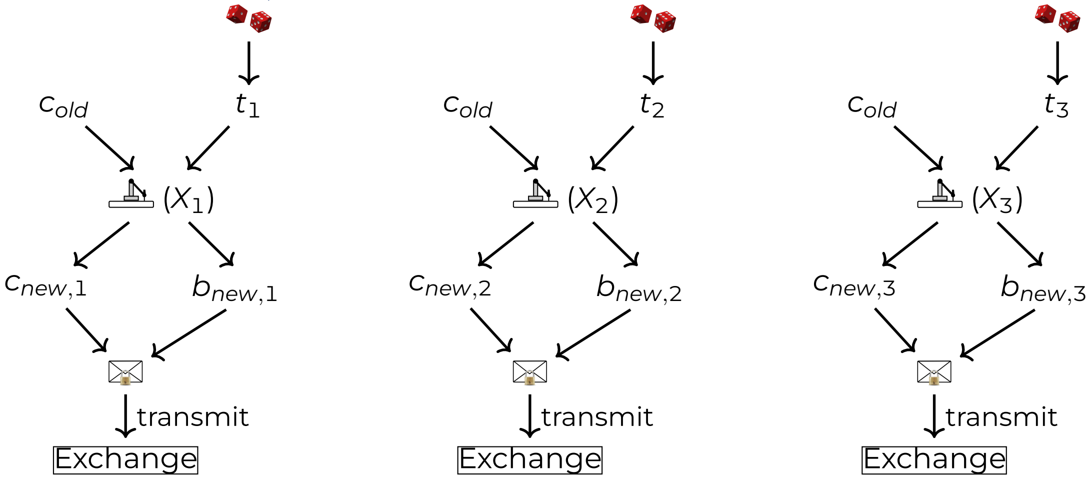
Customer: Transfer setup (UNISIG)

Given partially spent private coin key c_{old}:

1. Let $C_{old} := c_{old}G$ (as before)
2. Create random nonce t
3. Compute unique signature $X := UNISIG_{c_{old}}(t)$
4. Derive c_{new} and b_{new} from X using HKDF
5. Compute $C_{new} := c_{new}G$
6. Compute $f_{new} := FDH(C_{new})$
7. Transmit $f'_{new} := f_{new}b^{e}_{new}$



Cut-and-Choose



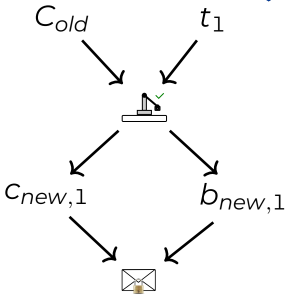
Exchange: Choose!

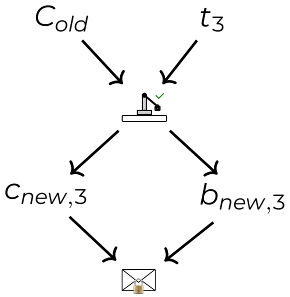
Exchange sends back random $\gamma \in \{1, 2, 3\}$ to the customer.

Customer: Reveal

1. If $\gamma = 1$, send $\langle t_2, X_2 \rangle$, $\langle t_3, X_3 \rangle$ to exchange
2. If $\gamma = 2$, send $\langle t_1, X_1 \rangle$, $\langle t_3, X_3 \rangle$ to exchange
3. If $\gamma = 3$, send $\langle t_1, X_1 \rangle$, $\langle t_2, X_2 \rangle$ to exchange

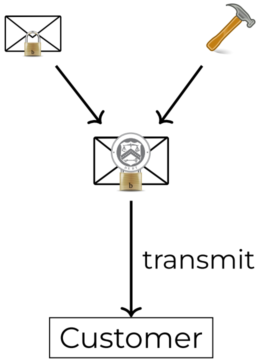
Exchange: Verify ($\gamma = 2$)





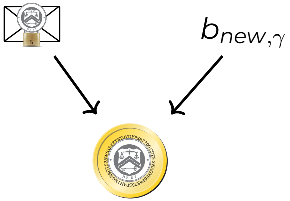
Exchange: Blind sign change (RSA)

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



Customer: Unblind change (RSA)

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

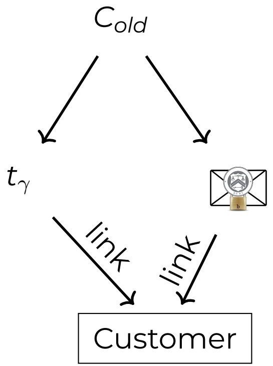


Exchange: Allow linking change

Given C_{old}

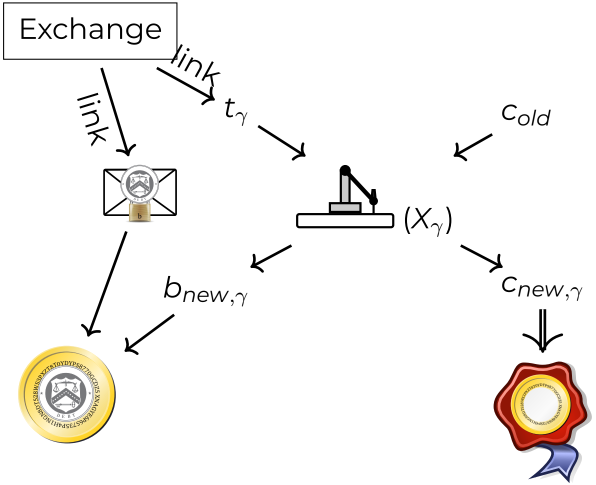
return t_γ and

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



Customer: Link (threat!)

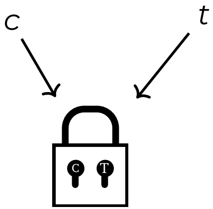
1. Have c_{old}.
2. Obtain T_γ, s from exchange
3. Compute $X_\gamma = UNISIG_{c_{old}}(t_\gamma)$
4. Derive $c_{new,\gamma}$ and $b_{new,\gamma}$ from X_γ
5. Unblind $s := s' b_{new,\gamma}^{-1} \mod n$



VRF vs. Dold'19 with Diffie-Hellman (ECDH)

VRF/unique signatures are *slightly* stronger than required!

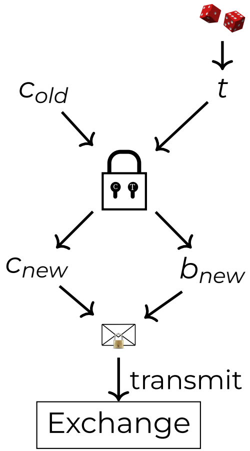
1. Create private keys $c, t \mod o$
2. Define $C = cG$
3. Define $T = tG$
4. Compute DH
   $cT = c(tG) = t(cG) = tC$
5. Sign T with EdDSA: DH is unique, with EdDSA we have a signature, t allows verifier to check!



Transfer setup with ECDH-based Refresh

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 $f'_{new} := f_{new}b^e_{new}$



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.

Component Zoo

The Taler Software Ecosystem: Overview

Taler is based on modular components that work together to provide a complete payment system:

- **Exchange:** Service provider for digital cash
  - Core exchange software (cryptography, database)
  - Air-gapped key management, real-time **auditing**
  - **libeufin**: Modular integration with banking systems
  - **challenger**: KYC service with OAuth 2.0 API
- **Merchant:** Integration service for existing businesses
  - Core merchant backend software (cryptography, database)
  - **Back-office interface** for staff
  - **Frontend integration** (E-commerce, Point-of-sale)
- **Wallet:** Consumer-controlled applications for e-cash
  - Multi-platform wallet software (for browsers & mobile phones)
  - Wallet backup storage providers (**sync** & **Anastasis**)

Taler Exchange

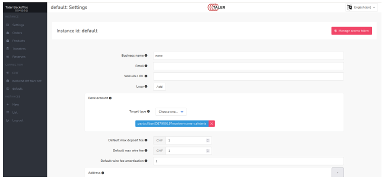
The **Exchange** is the core logic of the payment system.

- One exchange at minimum must be operated per currency
- Offers a REST API for merchants and customers
- Uses several helper processes for configuration and to interact with RTGS and cryptography
- KYC support via OAuth 2.0, KycAID or Persona APIs

Taler Merchant

The **Merchant** is the software run by merchants to accept GNU Taler payments.

- REST API for integration with e-commerce
- SPA provides Web interface for administration
- Features include:
  - Multi-tenant support
  - Refunds
  - Templates
  - Webhooks
  - Inventory management (optional)



default: Settings

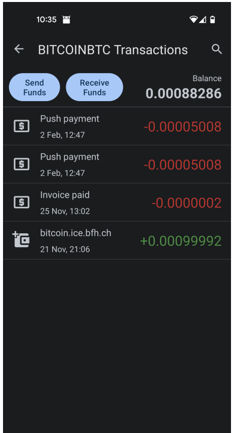
Instance id: default

Business name: none
Email:
Website URL:
Logo:
Bank account
Target type:
payto://iban/DE795919?receiver-name=cafeteria
Default max deposit fee: CHF 1
Default max wire fee: CHF 1
Default wire fee amortization: 1
Address

Taler Wallet

The **Wallet** is the software run by consumers to store their digital cash and authorize transactions.

- **wallet-core** is the logic shared by all interfaces
- Works on Android, F-Droid, iOS, Ubuntu Touch, WebExtension (Chrome, Chromium, Firefox, etc.)
- Features include:
  - Multi-currency support
  - Wallet-to-wallet payments (NFC or QR code)
  - CRDT-like data model


10:35

BITCOINBTC Transactions

Balance 0.00088286

Push payment 2 Feb, 12:47	-0.00005008
Push payment 2 Feb, 12:47	-0.00005008
Invoice paid 25 Nov, 13:02	-0.0000002
bitcoin.ice.bfh.ch 21 Nov, 21:06	+0.00099992

Taler Auditor

The **Auditor** is the software run by an independent auditor to validate the operation of an Exchange.

- REST API for additional report inputs by merchants (optional)
- Secure database replication logic

libeufin-nexus

libeufin-nexus allows Taler components to interact with a core banking system. It:

- provides an implementation of the Wire Gateway for the exchange
- supports EBICS 2.5 and 3.0
- other APIs such as FinTS or PSD2-style XS2A APIs can be added without requiring changes to the Exchange
- was tested with GLS Bank (DE) and Postfinance (CH) accounts and real EUR/CHF

libeufin-bank

libeufin-bank implements a standalone bank with a Web interface. It:

- provides the Taler Core Bank API for RESTful online banking using a Web interface (with multi-factor authentication)
- includes a Taler Wire Gateway for the exchange
- offers the Taler Bank Integration API to allow wallets to easily withdraw digital cash
- optionally provides the Taler Conversion Info API for currency conversion between fiat and regional currencies
- optionally integrates with libeufin-nexus to interact with a core banking system

Challenger

Challenger allows clients to obtain validated address (KYC) data about users:

- Customizable Web-based process for address validation
- Can validate phone numbers, e-mail addresses or physical mailing addresses
- Provides an exchange-compatible OAuth 2.0 API

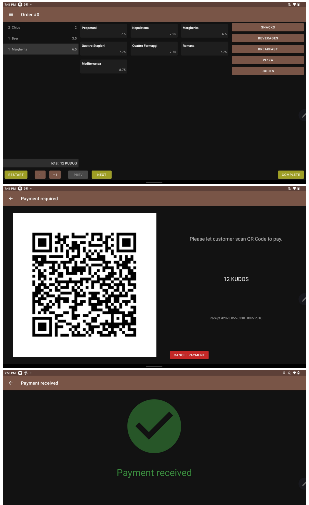
Depolymerization

Depolymerization is a bridge between GNU Taler and blockchains, making Taler a layer 2 system for crypto-currencies (like Lightning).

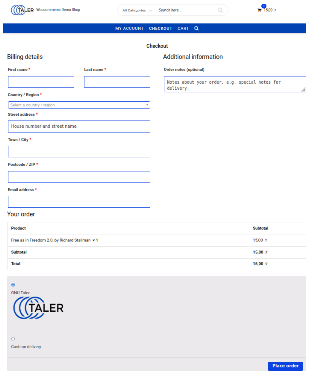
- provides an implementation of the Wire Gateway for the exchange
- Works on top of Bitcoin and Ethereum crypto-currencies, with the DLTs as the “RTGS”
- Provides same API to Exchange as libeufin-nexus

Point-of-Sale App for Android

- Allows merchant to generate orders against Taler backend and display QR code to enable customer to pay in person
- Patterned after ViewTouch restaurant UI



Payment plugins



Checkout

Billing details

First name *

Last name *

Country / Region *

Select a country / region...

Street address *

House number and street name

Town / City *

Postcode / ZIP *

Email address *

Additional information

Order notes (optional)

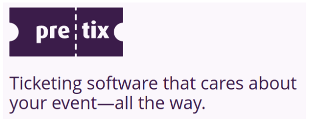
Notes about your order, e.g. special notes for delivery.

Your order

Product	Subtotal
Free as in Freedom 2.0, by Richard Stallman × 1	15,00
Subtotal	**15,00**
Total	**15,00**

[✓ GNU Taler]

Cash on delivery



Ticketing software that cares about your event—all the way.

- Pretix, ticket sales system
- Joomla!, an e-commerce platform
- WooCommerce, an e-commerce solution on top of WordPress
- DrupalCommerce, an e-commerce solution on top of Drupal

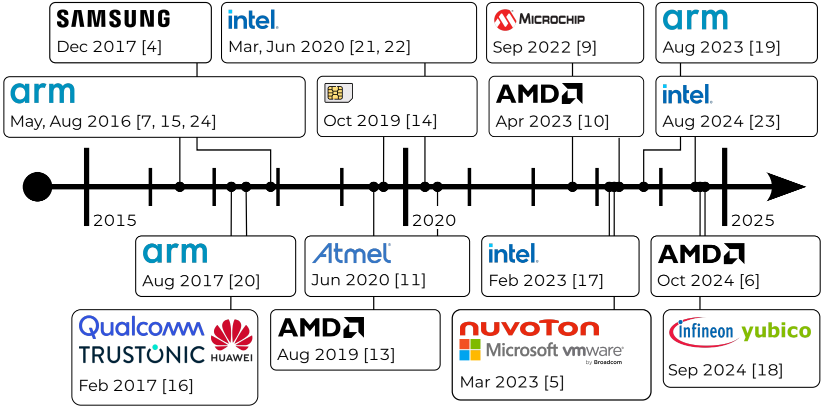
Offline payments

Digitaler Euro — Offline?

Many central banks today demand offline capabilities for CBDCs.

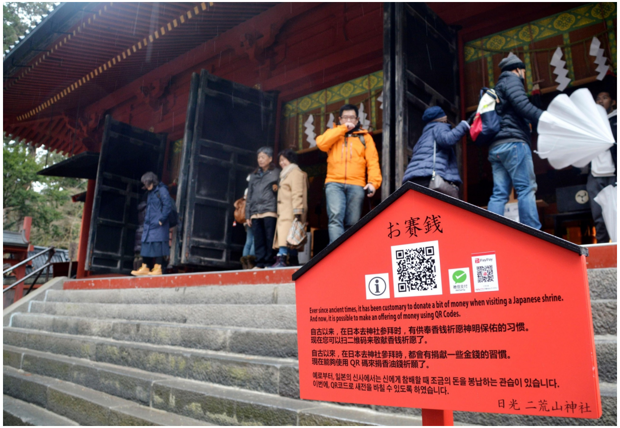
Digitaler Euro — Offline?

Many central banks today demand offline capabilities for CBDCs.



A Scenario

God is offline, but customer pays online

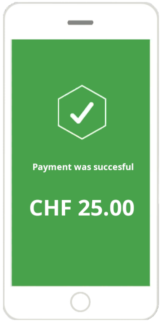

お賽銭
微信支付
WeChat Pay
PayPay
Ever since ancient times, it has been customary to donate a bit of money when visiting a Japanese shrine.
And now, it is possible to make an offering of money using QR Codes.
自古以来，在日本去神社参拜时，有供奉香钱祈愿神明保佑的习惯。
现在您可以扫二维码来敬献香钱祈愿了。
自古以來，在日本去神社參拜時，都會有捐獻一些金錢的習慣。
現在能夠使用 QR 碼來捐香油錢祈願了。
예로부터, 일본의 신사에서는 신에게 참배할 때 조금의 돈을 봉납하는 관습이 있습니다.
이번에, QR코드로 새전을 바칠 수 있도록 하였습니다.
日光 二荒山神社

Typical Payment Process

All equivalent: Twint, PayPal, AliPay, PayTM

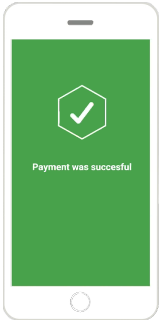
Secure Payment ...

Everything green?



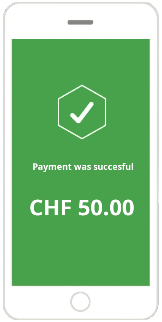
Exploit "Code"

Programming optional

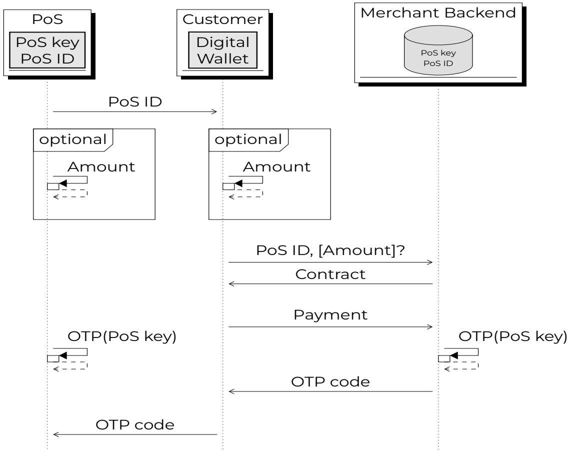


“Customers” *love* Twint …

Daily non-business for shops



Partially Offline Payments with GNU Taler [8]



Programmable money: Age restrictions [12]

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

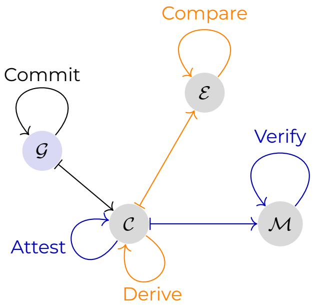
Age restriction
Assumptions and scenario

- 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



Formal Function Signatures

Searching for functions

- Commit
- Attest
- Verify
- Derive
- Compare

Formal Function Signatures

Searching for functions with the following signatures

Commit :	$(\mathsf{a}, \omega) \mapsto (\mathsf{Q}, \mathsf{P})$	$\mathbb{N}_{\mathsf{M}} \times \Omega \rightarrow \mathbb{O} \times \mathbb{P}$,
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

Commit :	$(a, \omega) \mapsto (Q, P)$	$\mathbb{N}_M \times \Omega \to \mathbb{O} \times \mathbb{P},$
Attest :	$(m, Q, P) \mapsto T$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \to \mathbb{T} \cup \{\perp\},$
Verify		
Derive		
Compare		

Mnemonics:
$\mathbb{O} = c\mathbb{O}mmitments$, $Q = Q\text{-}mitment$ (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

Commit :	$(a, \omega) \mapsto (Q, P)$	$\mathbb{N}_M \times \Omega \to \mathbb{O} \times \mathbb{P}$,
Attest :	$(m, Q, P) \mapsto T$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \to \mathbb{T} \cup \{\perp\}$,
Verify :	$(m, Q, T) \mapsto b$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{T} \to \mathbb{Z}_2$,
Derive		
Compare		

Mnemonics:
$\mathbb{O} = c\mathbb{O}mmitments$, $Q = Q\text{-}mitment$ (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

Commit :	$(a, \omega) \mapsto (Q, P)$	$\mathbb{N}_M \times \Omega \to \mathbb{O} \times \mathbb{P}$,
Attest :	$(m, Q, P) \mapsto T$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \to \mathbb{T} \cup \{\perp\}$,
Verify :	$(m, Q, T) \mapsto b$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{T} \to \mathbb{Z}_2$,
Derive :	$(Q, P, \omega) \mapsto (Q', P', \beta)$	$\mathbb{O} \times \mathbb{P} \times \Omega \to \mathbb{O} \times \mathbb{P} \times \mathbb{B}$,
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*, $\mathbb{B}$ = *$\mathbb{B}$lindings*, β = *βlinding*.

Formal Function Signatures

Searching for functions with the following signatures

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

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

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

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

Basic and security requirements are defined later.

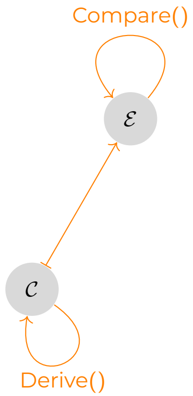
Mnemonics:
$\mathbb{O}$ = *c$\mathbb{O}$mmitments*, Q = *Q-mitment* (commitment), $\mathbb{P}$ = *$\mathbb{P}$roofs*, P = *Proof*,
$\mathbb{T}$ = *a$\mathbb{T}$testations*, T = *aTtestation*, $\mathbb{B}$ = *$\mathbb{B}$lindings*, β = *βlinding*.

Age restriction

Naïve scheme

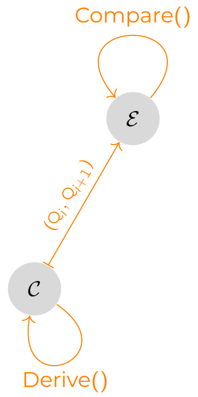
$$\begin{tikzcd}
{} & {} & \mathcal{E} \arrow[loop above, "\text{Compare}"] & {} \\
\mathcal{G} \arrow[loop above, "\text{Commit}"] \arrow[dr, mapsto] & {} & {} & {} \\
{} & \mathcal{C} \arrow[loop left, "\text{Attest}"] \arrow[loop below, "\text{Derive}"] \arrow[uur, mapsto] \arrow[rr, mapsto] & {} & \mathcal{M} \arrow[loop above, "\text{Verify}"]
\end{tikzcd}$$

Achieving Unlinkability



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

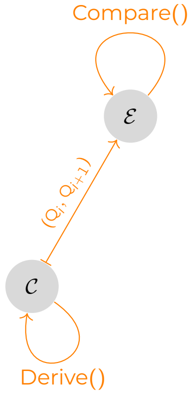
Achieving Unlinkability



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 $(Q_0, Q_1, \dots)$ 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 DeriveCompare$_\kappa$, using Derive() and Compare().

Sketch:

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

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

Achieving Unlinkability

With DeriveCompare$_\kappa$

- $\mathcal{E}$ learns nothing about Q$_\gamma$,
- 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

$$\begin{tikzcd}
 & & \mathcal{E} & \\
\mathcal{G} \arrow[loop above, "\mathrm{Commit}(a)"] \arrow[dr, "(Q, P_a)"'] & & & \\
 & \mathcal{C} \arrow[uur, leftrightarrow, "\mathrm{DeriveCompare}_{\kappa}"'] \arrow[rr, "(T_m, Q)"'] \arrow[loop below, "\mathrm{Attest}(m, Q, P_a)"] & & \mathcal{M} \arrow[loop below, "\mathrm{Verify}(m, Q, T_m)"]
\end{tikzcd}$$

Achieving Unlinkability

$\mathsf{DeriveCompare}_\kappa : \mathbb{O} \times \mathbb{P} \times \Omega \to \{0,1\}$
$\mathsf{DeriveCompare}_\kappa(\mathsf{Q},\mathsf{P},\omega) =$

```
$\mathcal{C}$:   1.  for all $i \in \{1,\dots,\kappa\}$ : $(\mathsf{Q}_i, \mathsf{P}_i, \beta_i) \leftarrow \mathsf{Derive}(\mathsf{Q}, \mathsf{P}, \omega + i)$
      2.  $h \leftarrow \mathsf{H}\big(\mathsf{H}(\mathsf{Q}_1, \beta_1) \parallel \cdots \parallel \mathsf{H}(\mathsf{Q}_\kappa, \beta_\kappa)\big)$
      3.  send $(\mathsf{Q}, h)$ to $\mathcal{E}$
$\mathcal{E}$:   4.  save $(\mathsf{Q}, h)$
      5.  $\gamma \xleftarrow{\$} \{1,\dots,\kappa\}$
      6.  send $\gamma$ to $\mathcal{C}$
$\mathcal{C}$:   7.  $h'_\gamma \leftarrow \mathsf{H}(\mathsf{Q}_\gamma, \beta_\gamma)$
      8.  $\mathbf{E}_\gamma \leftarrow \big[(\mathsf{Q}_1, \beta_1), \dots, (\mathsf{Q}_{\gamma-1}, \beta_{\gamma-1}), \perp, (\mathsf{Q}_{\gamma+1}, \beta_{\gamma+1}), \dots, (\mathsf{Q}_\kappa, \beta_\kappa)\big]$
      9.  send $(\mathbf{E}_\gamma, h'_\gamma)$ to $\mathcal{E}$
$\mathcal{E}$:  10.  for all $i \in \{1,\dots,\kappa\} \setminus \{\gamma\}$ : $h_i \leftarrow \mathsf{H}(\mathbf{E}_\gamma[i])$
     11.  if $h \overset{?}{\neq} \mathsf{H}(h_1 \| \dots \| h_{\gamma-1} \| h'_\gamma \| h_{\gamma+1} \| \dots \| h_{\kappa-1})$ return 0
     12.  for all $i \in \{1,\dots,\kappa\} \setminus \{\gamma\}$: if $0 \overset{?}{=} \mathsf{Compare}(\mathsf{Q}, \mathsf{Q}_i, \beta_i)$ return 0
     13.  return 1
```

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

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

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.

Requirements

Details

Derivability of commitments and proofs:

Let

$$\begin{gathered}\mathrm{a} \in \mathbb{N}_{\mathrm{M}},\ \omega_0, \omega_1 \in \Omega \\ (\mathrm{Q}_0, \mathrm{P}_0) \leftarrow \mathrm{Commit}(\mathrm{a}, \omega_0), \\ (\mathrm{Q}_1, \mathrm{P}_1, \beta) \leftarrow \mathrm{Derive}(\mathrm{Q}_0, \mathrm{P}_0, \omega_1).\end{gathered}$$

We require

$$\mathrm{Compare}(\mathrm{Q}_0, \mathrm{Q}_1, \beta) = 1$$

and for all $n \leq \mathrm{a}$:

$$\mathrm{Verify}(n, \mathrm{Q}_1, \mathrm{Attest}(n, \mathrm{Q}_1, \mathrm{P}_1)) = \mathrm{Verify}(n, \mathrm{Q}_0, \mathrm{Attest}(n, \mathrm{Q}_0, \mathrm{P}_0))$$

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

1. $(\mathsf{a}, \omega) \overset{\$}{\leftarrow} \mathbb{N}_{\mathsf{M}-1} \times \Omega$
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

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

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

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*

Solution: Instantiation with ECDSA

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

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

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

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

$$\Big\langle (q_1, p_1), \ldots, (q_a, p_a), (q_{a+1}, \bot), \ldots, (q_M, \bot) \Big\rangle$$

   - $\vec{Q} := (q_1, \ldots, q_M)$ is the *Commitment*,
   - $\vec{P}_a := (p_1, \ldots, p_a, \bot, \ldots, \bot)$ 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 $\vec{\mathsf{Q}} = (q_1, \ldots, q_{\mathsf{M}})$,
- (some) private-keys $\vec{\mathsf{P}} = (p_1, \ldots, p_{\mathsf{a}}, \perp, \ldots, \perp)$.

Instantiation with ECDSA

Definitions of Attest and Verify

Child has

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

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

Sign a message with ECDSA using private key p_m

Instantiation with ECDSA

Definitions of Attest and Verify

Child has

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

To Attest a minimum age $\mathrm{m} \leq \mathrm{a}$:
    Sign a message with ECDSA using private key p_m

Merchant gets

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

Instantiation with ECDSA

Definitions of Attest and Verify

Child has

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

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

Sign a message with ECDSA using private key p_m

Merchant gets

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

Instantiation with ECDSA

Definitions of Derive and Compare

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

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

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

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

Definitions of Derive and Compare

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

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

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

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

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

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

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

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.

Instantiation with ECDSA

Full definitions

$$\mathsf{Commit}_{E,[\cdot]_g}(\mathsf{a},\omega) := \left\langle \overbrace{(q_1,\ldots,q_\mathsf{M})}^{=\vec{\mathsf{Q}}}, \overbrace{(p_1,\ldots,p_\mathsf{a},\perp,\ldots,\perp)}^{=\vec{\mathsf{P}},\text{ length M}} \right\rangle$$

$$\mathsf{Attest}_{E,\mathsf{H}}(\mathsf{b},\vec{\mathsf{Q}},\vec{\mathsf{P}}) := \begin{cases} \mathsf{T}_\mathsf{b} := \mathsf{Sig}_{E,\mathsf{H}}(\mathsf{b},\vec{\mathsf{P}}[\mathsf{b}]) & \text{if } \vec{\mathsf{P}}[\mathsf{b}] \overset{?}{\neq} \perp \\ \perp & \text{otherwise} \end{cases}$$

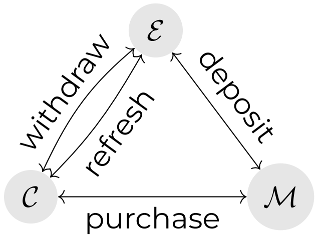
$$\mathsf{Verify}_{E,\mathsf{H}}(\mathsf{b},\vec{\mathsf{Q}},\mathsf{T}) := \mathsf{Ver}_{E,\mathsf{H}}(\mathsf{b},\vec{\mathsf{Q}}[\mathsf{b}],\mathsf{T})$$

$$\mathsf{Derive}_{E,[\cdot]_g}(\vec{\mathsf{Q}},\vec{\mathsf{P}},\omega) := \left\langle (\beta * q_1,\ldots,\beta * q_\mathsf{M}),(\beta p_1,\ldots,\beta p_\mathsf{a},\perp,\ldots,\perp),\beta \right\rangle$$

$$\text{with } \beta := [\omega]_g \text{ and multiplication } \beta p_i \text{ modulo } g$$

$$\mathsf{Compare}_E(\vec{\mathsf{Q}},\vec{\mathsf{Q}}',\beta) := \begin{cases} 1 & \text{if } (\beta * q_1,\ldots,\beta * q_\mathsf{M}) \overset{?}{=} (q'_1,\ldots,q'_\mathsf{M}) \\ 0 & \text{otherwise} \end{cases}$$

Reminder: GNU Taler Fundamentals


$$\begin{tikzcd} & \mathcal{E} \arrow[dl, bend right, "withdraw"'] \arrow[dr, "deposit"] & \\ \mathcal{C} \arrow[ur, bend right, "refresh"'] \arrow[rr, leftrightarrow, "purchase"'] & & \mathcal{M} \end{tikzcd}$$

- 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
Binding age restriction to coins

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

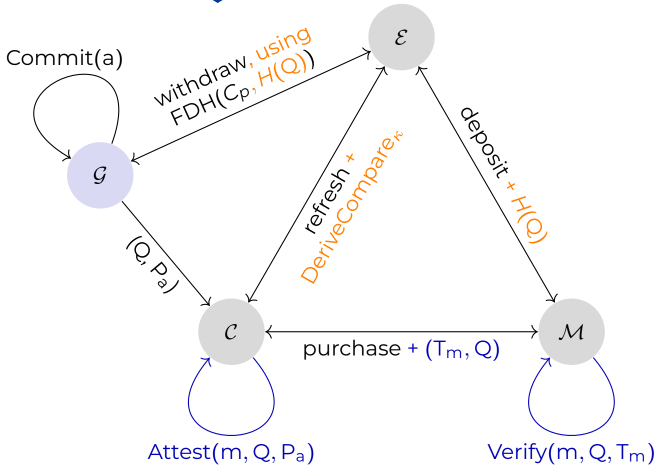
$$\mathrm{FDH}(C_p, H(\mathrm{Q}))$$

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

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

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.

Age Restrictions based on KYC

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.

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

Future Work & Conclusion

Use Case: Journalism

Today:

- Corporate structure
- Advertising primary revenue
- Tracking readers critical for business success
- Journalism and marketing hard to distinguish

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

Taler: Project Status

`https://docs.taler.net/`

- Cryptographic protocols and core exchange component are stable
- Pilot project at Bern University of Applied Sciences cafeteria
- Netzbon (regional currency) in Basel launched
- Taler Operations AG live Swiss-wide
- Internal alpha deployment with GLS Bank (Germany)
- Internal alpha deployment with Magnet Bank (Hungary)

Competitor comparison

	Cash	Bitcoin	Zerocoin	Creditcard	GNU Taler
Online	−−−	++	++	+	+++
Offline	+++	−−	−−	+	++
Trans. cost	+	−−−	−−−	−	++
Speed	+	−−−	−−−	o	++
Taxation	−	−−	−−−	+++	+++
Payer-anon	++	o	++	−−−	+++
Payee-anon	++	o	++	−−−	−−−
Security	−	o	o	−−	++
Conversion	+++	−−−	−−−	+++	+++
Libre	−	+++	+++	−−−	+++

Other ongoing developments

- Privacy-preserving auctions (trading, currency exchange) (`oezguer@taler.net`)
- Hardware and software support for embedded systems (`mikolai@taler.net`)
- Tax-deductable receipts for donations to charities (donau.git)
- Unlinkable anonymous subscriptions and discount tokens (`ivan@taler.net`)
- Support for illiterate and innumerate users[1] (`marc@taler.net`)

[1] Background: `https://myoralvillage.org/`

Open Challanges

- Try to explain this to lawyers and AML staff of banks
- What are convincing arguments for citizens to switch?
- How to address anti-competitive cash-back from card payments?
- . . .

How to support?

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

Discuss: `https://ich.taler.net/`

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/`

Conclusion

What can we do?

- Suffer mass-surveillance enabled by credit card oligopolies with high fees, and
- Engage in arms race with deliberately unregulatable blockchains

OR

- Establish free software alternative balancing social goals!

References I

Jeffrey Burdges, Florian Dold, Christian Grothoff, and Marcello Stanisci.
Enabling secure web payments with GNU Taler.
In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *6th International Conference on Security, Privacy and Applied Cryptographic Engineering*, number 10076 in LNCS, pages 251–270. Springer, Dec 2016.

David Chaum, Christian Grothoff, and Thomas Moser.
How to issue a central bank digital currency.
In *SNB Working Papers*, number 2021-3. Swiss National Bank, February 2021.

References II

Florian Dold.
The GNU Taler system: practical and provably secure electronic payments. (Le système GNU Taler: Paiements électroniques pratiques et sécurisés).
PhD thesis, University of Rennes 1, France, 2019.

M. Dorjmyagmar, M. Kim, and H. Kim.
Security analysis of samsung knox.
In *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 550–553, 2017.

References III

Francisco Falcon.
Vulnerabilities in the tpm 2.0 reference implementation code.
https://blog.quarkslab.com/vulnerabilities-in-the-tpm-20-reference-implementation-code.html,
March 2023.

References IV

Stefan Gast, Hannes Weissteiner, Robin Leander Schröder, and Daniel Gruss.
Counterseveillance: Performance-counter attacks on amd sev-snp.
In *Network and Distributed System Security (NDSS) Symposium 2025*, February 2025.
Network and Distributed System Security Symposium 2025 : NDSS 2025, NDSS 2025 ; Conference date: 23-02-2025 Through 28-02-2025.

References V

R. Guanciale, H. Nemati, C. Baumann, and M. Dam.
Cache storage channels: Alias-driven attacks and verified countermeasures.
In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 38–55, May 2016.

Priscilla Huang, Emmanuel Benoist, Christian Grothoff, and Sebastian Javier Marchano.
Practical offline payments using one-time passcodes.
SUERF Policy Briefs, (622), June 2023.

References VI

Olivier Hériveaux.
Triple exploit chain with laser fault injection on a secure element.
In *2022 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)*, pages 9–17, 2022.

Hans Niklas Jacob, Christian Werling, Robert Buhren, and Jean-Pierre Seifert.
faultpm: Exposing amd ftpms' deepest secrets.
In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 1128–1142. IEEE, 2023.

References VII

- Jan Jancar, Vladimir Sedlacek, Petr Svenda, and Marek Sys.
  Minerva: The curse of ECDSA nonces (systematic analysis of lattice attacks on noisy leakage of bit-length of ECDSA nonces).
  IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(4):281–308, 2020.

- Özgür Kesim, Christian Grothoff, Florian Dold, and Martin Schanzenbach.
  Zero-Knowledge Age Restriction for GNU Taler.
  In Vijayalakshmi Atluri, Roberto Di Pietro, Christian D. Jensen, and Weizhi Meng, editors, *Computer Security – ESORICS 2022*, pages 110–129, Cham, 2022. Springer International Publishing.

References VIII

- Mengyuan Li, Yinqian Zhang, Zhiqiang Lin, and Yan Solihin.
  Exploiting unprotected i/o operations in amd's secure encrypted virtualization.
  In *USENIX Security Symposium*, 2019.
- Adaptive Mobile Security Limited.
  Simjacker technical report.
  `https://www.enea.com/info/simjacker/`, 2019.

References IX

Moritz Lipp, Daniel Gruss, Raphael Spreitzer, Clémentine Maurice, and Stefan Mangard.
Armageddon: Cache attacks on mobile devices.
In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 549–564, USA, 2016. USENIX Association.

Aravind Machiry, Eric Gustafson, Chad Spensky, Christopher Salls, Nick Stephens, Ruoyu Wang, Antonio Bianchi, Yung Ryn Choe, Christopher Kruegel, and Giovanni Vigna.
Boomerang: Exploiting the semantic gap in trusted execution environments.
In *NDSS*, 2017.

References X

- Joseph Nuzman.
  Cve-2022-38090: Improper isolation of shared resources in some intel(r) processors when using intel(r) software guard extensions may allow a privileged user to potentially enable information disclosure via local access.
  `https://www.cve.org/CVERecord?id=CVE-2022-38090`, February 2023.
- Thomas Roche.
  Eucleak: Side-channel attack on the yubikey 5 series—revealing and breaking infineon ecdsa implementation on the way.
  `https://ninjalab.io/eucleak/`, September 2024.

References XI

- Xhani Marvin Saß, Richard Mitev, and Ahmad-Reza Sadeghi.
  Oops..! i glitched it again! how to Multi-Glitch the Glitching-Protections on ARM TrustZone-M.
  In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6239–6256, 2023.
- Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo.
  Clkscrew: Exposing the perils of security-oblivious energy management.
  In *Proceedings of the 26th USENIX Conference on Security Symposium*, SEC'17, page 1057–1074, USA, 2017. USENIX Association.

References XII

- Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lipp, Marina Minkin, Daniel Genkin, Yarom Yuval, Berk Sunar, Daniel Gruss, and Frank Piessens.
  LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection.
  In *41th IEEE Symposium on Security and Privacy (S&P'20)*, March 2020.
- Stephan van Schaik, Andrew Kwong, Daniel Genkin, and Yuval Yarom.

  SGAxe: How SGX fails in practice.
  `https://sgaxeattack.com/`, June 2020.

References XIII

Luca Wilke, Florian Sieck, and Thomas Eisenbarth.
Tdxdown: Single-stepping and instruction counting attacks against intel tdx.
In *ACM CCS 2024*, 2024.

Ning Zhang, Kun Sun, Deborah Shands, Wenjing Lou, and Y Thomas Hou.
Truspy: Cache side-channel information leakage from the secure world on arm devices.
IACR Cryptol. ePrint Arch., 2016:980, 2016.

Acknowledgments

Funded by the European Union (Project 101135475).

Co-funded by
the European Union

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