NEXT GENERATION INTERNET
GNU Taler

Christian Grothoff

07.06.2024

NEXT GENERATION INTERNET
GNU Taler
Christian Grothoff
07.06.2024

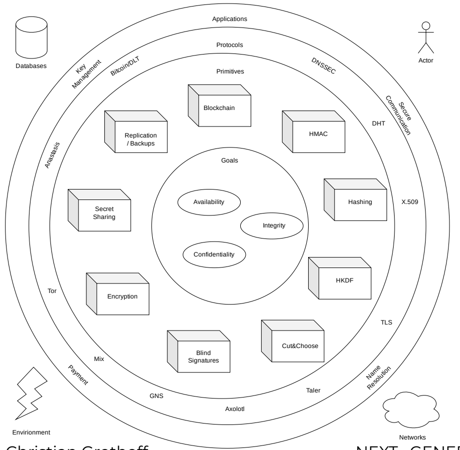
1. In this lecture, we will do a deep dive into the cryptography behind GNU Taler.

Learning objectives

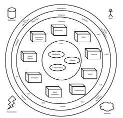
How should we pay?
Introduction to GNU Taler
How does cut-and-choose work?
How to prove protocols secure with cryptographic games?
What are the future plans for GNU Taler?

1. First, following Prof. Rogaway's call [5] for a community-wide effort to develop more effective means to resist mass surveillance, we will start with a moral analysis of the problem: How should we pay?
2. Then, you will get an overview of GNU Taler and its architecture. You should already be familiar with blind signatures, one key cryptographic building block.
3. We will then use GNU Taler's cryptography to introduce two more advanced cryptographic concepts, namely cut-and-choose protocols and an advanced example for provable security using cryptographic games.
4. Finally, we'll take a brief peek at the GNU Taler roadmap to see what probably lies ahead.

Taler on our map



NGI TALER



1. Taler is a payment protocol offering anonymity.
2. Key building blocks are blind signatures and cut&choose constructions.
3. The main related topics are Tor (anonymity), Blockchain (payment) and Integration (payments usually need to be integrated with other business processes).

How should we pay?

How should we pay?

Surveillance

1. What domain of digital communication should we be most concerned about?

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

- 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

1. Internet mass-surveillance may be bad, but it is to some degree avoidable or escapable.

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?"*

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?"

1. The result: an electronic funds transfer system that looks strikingly similar today's debit card system.
2. What is surprising is that Snowden says this is **one of the worst things**, as he obviously had a bunch of rather large concerns.

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)

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)

1. The result: an electronic funds transfer system that looks strikingly similar today's debit card system.
2. What is surprising is that Snowden says this is **one of the worst things**, as he obviously had a bunch of rather large concerns.

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!

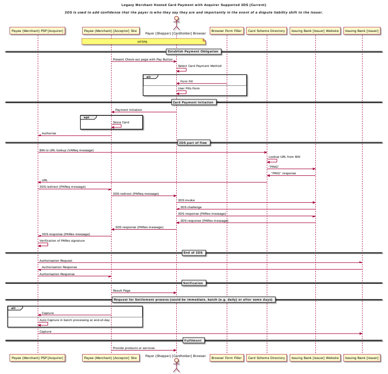
- When you pay by CC, the information includes your name
- When you pay in person with CC, your location is also known
- You often have no alternative payment methods available
- You hardly ever can use someone else's CC
- Anonymous prepaid cards are difficult to get and expensive
- Payment information is typically stored for 6-10 years!

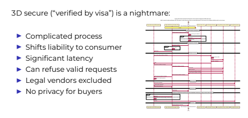
1. For digital payments, surveillance has become completely normalized.
2. It is also basically inescapable, except by using cash. and cash sometimes cannot be used!

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





1. Now, the modern online CC process is also a nightmare, from privacy, security, usability and cost perspectives.
2. Our claim: online credit card payments will be replaced. The question is, with what?

The bank's Problem

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

1. And Apple would like us to pay 30% fees on everything for their walled surveillance garden.

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

- 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

1. "Do you want to live under total surveillance?" Sure, this may sound unlikely, but let's listen to some experts on this.

The Bank of International Settlements on CBDC

The Bank of International Settlements on CBDC

1. The key sentence here is that they **will have absolute control** over how we use digital cash.
2. The director of the BIS points to this as a fact. Note that the BIS is basically the United Nations of the central banks of the world. Their headquarters is in Basel.

The Emergency Act of Canada, February 2022, https://www.youtube.com/watch?v=Neb

The Emergency Act of Canada, February 2022, https://www.youtube.com/watch?v=Neb

Introduction to GNU Taler

Introduction to GNU Taler

GNU Taler

Digital cash, made **socially responsible**.

TALER

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

Digital cash, made **socially responsible**.
TALER
Privacy-Preserving, Practical, Taxable, Free Software, Efficient

1. Bold claims.

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”

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”

1. Not a currency, not a crypto-currency, no blockchain, just a payment system.
2. For your day-to-day expenses, not your retirement fund or buying a house.
3. Not like PayPal where you have one operator, primarily a protocol, like HTTP!
4. Not a P2P network, there are still easily identifiable (and accountable) payment service providers.
5. Efficient, no burning down the planet for 3 transactions per second.
6. Again, not a currency, you pay with GNU Taler, not *in* Taler. You pay in EUR/CHF/USD.

Design goals

... for the GNU Taler payment system

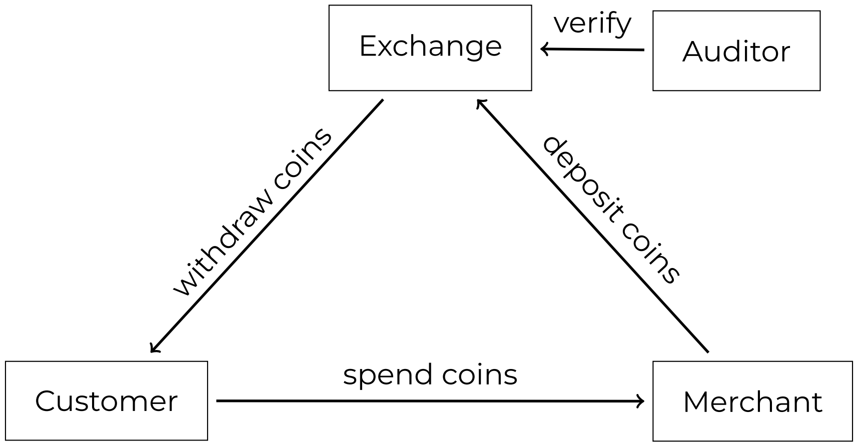
GNU Taler must ...

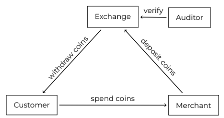
1. ... be implemented as **free software**.
2. ... protect the **privacy of buyers**.
3. ... must enable the state to **tax income** and crack down on illegal business activities.
4. ... prevent payment fraud.
5. ... only **disclose the minimal amount of information necessary**.
6. ... be usable.
7. ... be efficient.
8. ... avoid single points of failure.
9. ... foster **competition**.

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**.

1. The design goals are presented in order of priority. Those higher up are more important.
2. If you have other priorities, you will end up with a different design. When designing a system, try to come up with priorities first.
3. Of course the order is not absolute: we would not sacrifice an insane amount of efficiency for a tiny gain in usability. But it is important to have priorities when the trade-offs are plausible.
4. Objective 5 is relevant as objective 2 is only about privacy of buyers, but there is other data to minimize in a complex system.
5. How do you foster competition? By making it possible for various components to be commercially offered by different parties; using proper protocols ensures there can be different implementations, operators and integrators.

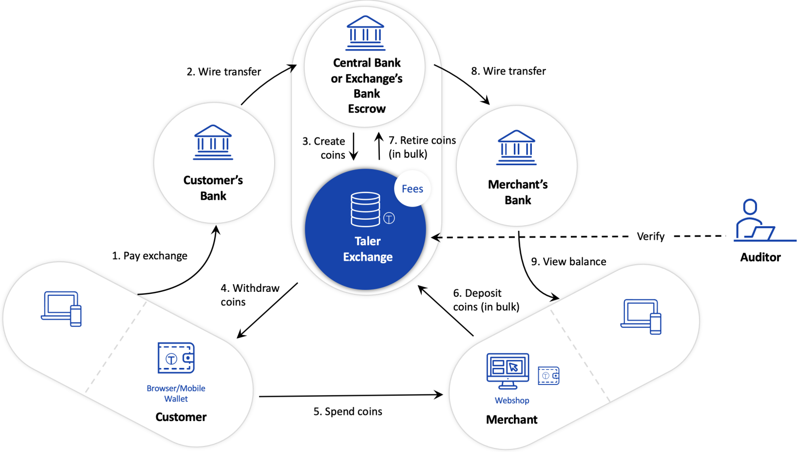
Taler overview





1. This figure shows the key parties in the GNU Taler system.
2. The exchange operates the payment system: it issues digital coins and allows them to be redeemed.
3. Customers obtain digital cash can can spend it.
4. Merchants accept digital cash.
5. The auditor checks that the exchange is operating correctly.

Architecture of Taler



1. Illustration of the payment process of GNU Taler and its integration with an existing core banking system. Macro-payments between bank accounts (steps 2 and 8) are for large sums. Step 2 represents buyers withdrawing money from their bank accounts, and step 8 merchants receiving their aggregated (daily, weekly, monthly, etc.) revenues. In contrast, cryptographic payments within GNU Taler (steps 4, 5 and 6) are much cheaper.
2. The purchase in step 5 is unlinkable to the withdrawal in step 4 due to the use of blind signatures, which protect the anonymity for the buyer spending coins in step 5.

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.

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.

1. KUDOS is a “fake” currency used for the demonstration, EUR or CHF would be used in practice.
2. The demo can be done using a WebExtension or a Taler wallet running on a mobile phone, or both.
3. You can also demonstrate P2P payments between the Taler wallet browser extension and the mobile phone.

How does cut-and-choose work?

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.

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.

1. When withdrawing, Taler does not exactly determine that the owner of the account is also the owner of the wallet. The withdraw loophole is basically equivalent to somebody putting their bank card into an ATM and someone else taking the cash. While [3] explains a way to address the loophole, doing so would put the privacy of payer's at risk, so we decided against it and will not cover it here.
2. The sharing loophole is when the owner of a coin decides to simply give the private key and signature of a digital coin to another user. As the user cannot be sure that the owner really deleted their copy of the private key material (without any backup), both users then *share* access to the value of the coin. The first to spend it, will succeed. We call this *sharing* as opposed to a transaction: in a transaction, ownership is transferred between parties that do not trust each other.
3. Sharing is thus like giving your spouse the password to your bank account.

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.

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.

1. Taler issues digital cash using blind signatures, where each signature conveys the respective coin a particular value.
2. We want to avoid cryptographic expenses linear in the amount being paid!
3. Thus we need a way to get change, but doing so must not void our security assurances, specifically unlinkability (and anonymity) for the payer, and income transparency for the payee.
4. The high-level approach for getting change is pretty simple: when paying with a coin, the (EdDSA) coin signature can specify that not the full value of the coin is to be spent, but only a fraction. The exchange then allows a wallet to request change by creating a second signature using the partially spent coin's private (EdDSA) key over a change request with fresh (blinded) digital coins that total up to the amount of change that is due.

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

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

1. Taler issues digital cash using blind signatures, where each signature conveys the respective coin a particular value.
2. We want to avoid cryptographic expenses linear in the amount being paid!
3. Thus we need a way to get change, but doing so must not void our security assurances, specifically unlinkability (and anonymity) for the payer, and income transparency for the payee.
4. The high-level approach for getting change is pretty simple: when paying with a coin, the (EdDSA) coin signature can specify that not the full value of the coin is to be spent, but only a fraction. The exchange then allows a wallet to request change by creating a second signature using the partially spent coin's private (EdDSA) key over a change request with fresh (blinded) digital coins that total up to the amount of change that is due.

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:

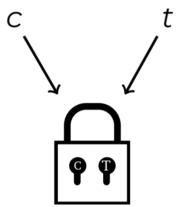
- Contract can specify to only pay *partial value* of a coin.
- Exchange allows wallet to obtain *unlinkable change* for remaining coin value.

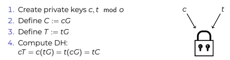
It would be inefficient to pay EUR 100 with 1 cent coins!
- Denomination key represents value of a coin.
- Exchange may offer various denominations for coins.
- Wallet may not have exact change!
- Usability requires ability to pay given sufficient total funds.
Key goals:
- maintain unlinkability
- maintain 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.

1. Taler issues digital cash using blind signatures, where each signature conveys the respective coin a particular value.
2. We want to avoid cryptographic expenses linear in the amount being paid!
3. Thus we need a way to get change, but doing so must not void our security assurances, specifically unlinkability (and anonymity) for the payer, and income transparency for the payee.
4. The high-level approach for getting change is pretty simple: when paying with a coin, the (EdDSA) coin signature can specify that not the full value of the coin is to be spent, but only a fraction. The exchange then allows a wallet to request change by creating a second signature using the partially spent coin's private (EdDSA) key over a change request with fresh (blinded) digital coins that total up to the amount of change that is due.

Diffie-Hellman (ECDH)

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





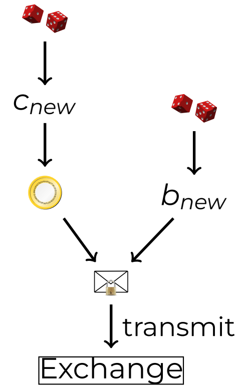
1. Before we can introduce the change protocol, we need another pretty picture for a well-known cryptographic primitive, Diffie-Hellman (DH). Taler uses ECDH, but that does not matter except for performance.
2. A good way to think of DH is a lock with two keys where either key opens the lock.
3. Note that we will use DH in a rather unusual way in the following protocol.

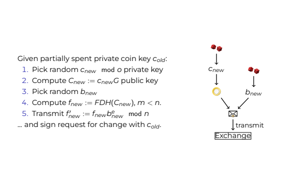
Straw-man solution

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}.



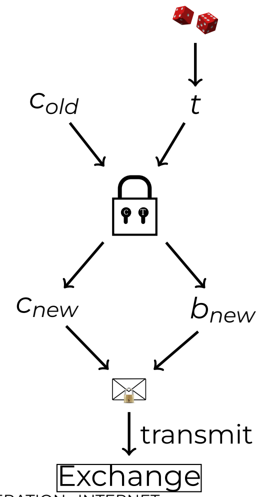


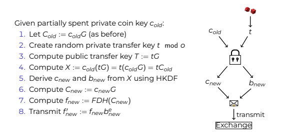
1. A straw-man solution is one that does not work, but still could be useful to illuminate the issue.
2. Here, the protocol allows users to obtain change (c_{new}) by signing the request for change (the envelope) with an old coin c_{old} that has some residual value from a previous purchase (that signature is not shown).
3. **Problem**: Owner of c_{new} may differ from owner of c_{old} breaks income-transparency / enables tax evasion!

Customer: Transfer key setup (ECDH)

Given partially spent private coin key c_{old}:

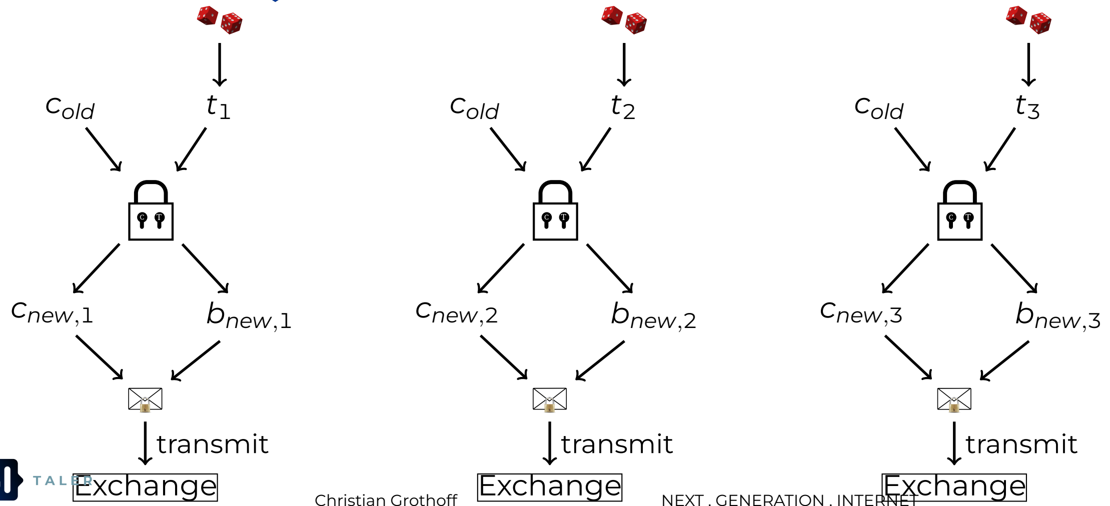
1. Let $C_{old} := c_{old}G$ (as before)
2. Create random private transfer key $t \mod o$
3. Compute public transfer key $T := tG$
4. Compute $X := c_{old}(tG) = t(c_{old}G) = tC_{old}$
5. Derive c_{new} and b_{new} from X using HKDF
6. Compute $C_{new} := c_{new}G$
7. Compute $f_{new} := FDH(C_{new})$
8. Transmit $f'_{new} := f_{new}b^e_{new}$

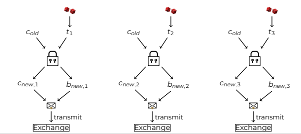




1. In this construction, we *derive* the blinding factor b_{new} and the private key of the new coin c_{new} from the DH of the c_{old} and a newly created transfer key t. Note that it is a bit unusual but perfectly find that we here have **both** private keys to compute the DH.
2. The resulting blinded public key of the new coin (public key derivation and blinding are elided to keep the diagram concise) is then signed with c_{old} to request change.
3. This approach has an obvious problem: from the perspective of the Exchange, we cannot even tell that the user followed this procedure as the resulting request with the blinded coin is indistinguishable from the previous construction.

Cut-and-Choose





1. This DH-construction thus obviously does not work, so in the usual approach of an insane person, we don't just do it once, but three times using three different transfer keys t_1, t_2, and t_3 instead of just t.
2. Now, before you decide that we have just gone mad, this is actually a well-known technique called **cut-and-choose**. Here, we do a protocol step multiple times to basically be able to **burn** some of these iterations to **prove** our honesty.
3. There are also **non-interactive** cut-and-choose protocols, but this one is a simple interactive one.

Exchange: Choose!

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

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

1. This is the typical interaction: the Exchange picks one of the three at random, basically deciding on which iterations to challenge the wallet's honesty.
2. γ primarily needs to be **unpredictable** for the wallet.
3. Note that the protocol has a security parameter $\kappa = 3$, and so the wallet could guess correctly in $\frac{1}{3}$ of the cases. Usually in security we would think of this to be way too low, and you will see much higher values in other cut-and-choose protocols. But, we will see why $\kappa = 3$ is actually enough for GNU Taler!

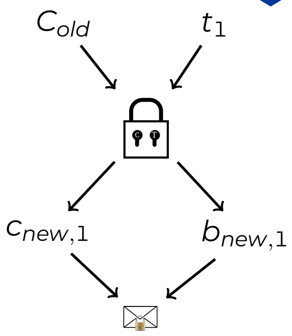
Customer: Reveal

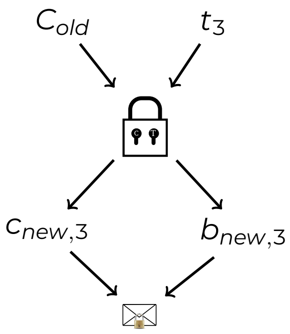
1. If $\gamma = 1$, send t_2, t_3 to exchange
2. If $\gamma = 2$, send t_1, t_3 to exchange
3. If $\gamma = 3$, send t_1, t_2 to exchange

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

1. So given the γ challenge value, the wallet has to send back the t_i values for $i \neq \gamma$.

Exchange: Verify ($\gamma = 2$)

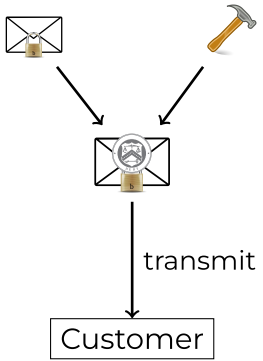


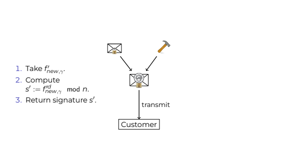


1. Given those two values the exchange can **validate** the construction as it can compute the DH from the **transfer private keys** t_i and the **coin public key** C_{old}.
2. If the result matches with the original request from the wallet, the exchange has established that with $\frac{2}{3}$ probability the wallet made an honest request for change following the prescribed construction.
3. If the wallet is unable (or unwilling) to produce the required t_i values, or if the resulting blinded values do not match, the entire change is forfeit, and the customer looses their money.
4. Thus, trying to cheat on income-transparency is punished with what amounts to a **66.67% tax**. Thus, a security level of κ is sufficient as long as the *effective* income tax (after deductions, on the full income) is below $\frac{\kappa-1}{\kappa}$. Taler always uses $\kappa = 3$.

Exchange: Blind sign change (RSA)

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

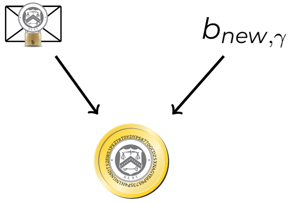


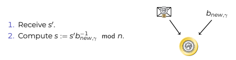


1. If the customer's request did follow the DH-construction, the exchange takes the third envelope, the one where t_γ was not disclosed, and signs this one to issue the change.

Customer: Unblind change (RSA)

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





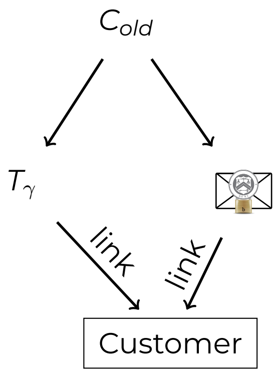
1. As with the ordinary blind-signature based withdraw, the customer can then unblind the signature and has a valid coin.
2. Without knowledge of c_{old} or t_γ, the coins derived from this process are indistinguishable from coins that were withdrawn directly from an account.
3. Most importantly, without knowledge of t_γ or c_{old}, the c_{new} is unlinkable to c_{old}.

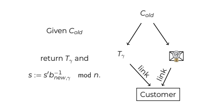
Exchange: Allow linking change

Given C_{old}

return T_γ and

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

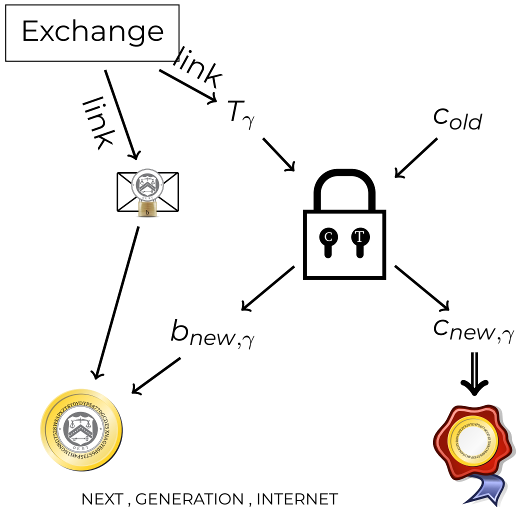


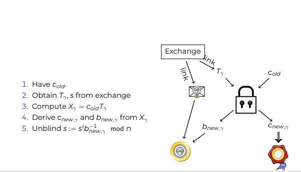


1. But, how does this address the issue that c_{old} may have a different owner from $c_{new,\gamma}$? Well, so far it does not! In principle, the envelope can easily be constructed by someone who was not the original owner of c_{old}.
2. So how does this help? Well, the exchange has one more sub-protocol, which is the **link** protocol. Given the old coin's public key, C_{old}, it returns T_γ, the **public transfer key**, and the blind signature over the new coin that was rendered as change.
3. Note that this is a request that the owner of c_{old} can always trivially make, as they know C_{old}.
4. So how does that help?

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





1. Well, given these two values, the owner of the original c_{old} can **also** compute the DH (this time from c_{old} and T_γ), and then also derive $c_{new,\gamma}$ and also unblind the exchange's signature using $b_{new,\gamma}$.
2. As a result, the owner of the old coin can always compute the change, and thus is effectively **also** always an owner of the change rendered!
3. Thus, we have **reduced** the possibility of abusing the change protocol for a transaction that would result in a **mutually exclusive transfer of ownership** to the case where the ownership of the change is **shared**.
4. But, we previously explained that **sharing** is not something we can or would care to prevent, so the change protocol does not weaken income transparency.

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.

- 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.

1. In Taler, the overall protocol is called the **refresh** protocol, not the **change** protocol, as it has uses beyond getting unlinkable change.
2. A merchant can grant a refund to an anonymous customer by telling the exchange to nullify the original deposit. Then the anonymous owner of the original coin can obtain the refund via the refresh protocol.
3. If a coin is about to expire (because the exchange only accepts deposits for a certain denomination key for a limited amount of time), the refresh protocol can be used to obtain fresh coins, signed with the current denomination key. This is like rolling over to a fresh series of bank notes.
4. Finally, we can handle situations where the customer did try to spend digital cash, but then the message was lost, say due to a power outage, before the transaction was actually completed. But, the customer might not be sure that nobody else saw the public key of the coin! So, to ensure that transactions remain unlinkable (and that the merchant cannot deposit the coin later), the wallet can again use the refresh protocol.

How to prove protocols secure with cryptographic games?

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 $i \in \{1, \ldots, q\}$.
- **Oracle** Given M_0 and M_1 (of same length), return $C := \texttt{enc}(k, M_b)$.

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.

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 $i \in \{1, \ldots, q\}$.
Oracle Given M_0 and M_1 (of same length), return $C := \texttt{enc}(k, M_b)$.
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

Problem:
Verification of minimum age requirements in e-commerce.
Common solutions:
1. ID Verification
2. Restricted Accounts
3. Attribute-based

1. *Restricted Accounts* are for example Credit or Debit cards specific for children. Such cards exist e.g. in the US and only allow to purchase in certain stores. More generally, Debit cards are sometimes also used as proxy to proof that an adult is doing a purchase, e. g. when buying cigarettes at a vending machine with cash money.
2. *Attribute-based* systems that are operational are e.g. the IRMA app in the Netherlands. Another (not widespread) example is re:claimid, work done by the GNUnet e.V.,

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

Problem:
Verification of minimum age requirements in e-commerce.
Common solutions:
Privacy
1. ID Verification bad
2. Restricted Accounts bad
3. Attribute-based good

1. *Restricted Accounts* are for example Credit or Debit cards specific for children. Such cards exist e.g. in the US and only allow to purchase in certain stores. More generally, Debit cards are sometimes also used as proxy to proof that an adult is doing a purchase, e. g. when buying cigarettes at a vending machine with cash money.
2. *Attribute-based* systems that are operational are e.g. the IRMA app in the Netherlands. Another (not widespread) example is re:claimid, work done by the GNUnet e.V.,

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

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

1. *Restricted Accounts* are for example Credit or Debit cards specific for children. Such cards exist e.g. in the US and only allow to purchase in certain stores. More generally, Debit cards are sometimes also used as proxy to proof that an adult is doing a purchase, e. g. when buying cigarettes at a vending machine with cash money.
2. *Attribute-based* systems that are operational are e.g. the IRMA app in the Netherlands. Another (not widespread) example is re:claimid, work done by the GNUnet e.V.,

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

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

1. *Restricted Accounts* are for example Credit or Debit cards specific for children. Such cards exist e.g. in the US and only allow to purchase in certain stores. More generally, Debit cards are sometimes also used as proxy to proof that an adult is doing a purchase, e. g. when buying cigarettes at a vending machine with cash money.
2. *Attribute-based* systems that are operational are e.g. the IRMA app in the Netherlands. Another (not widespread) example is re:claimid, work done by the GNUnet e.V.,

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*.

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*.

Historically, the concept of *subsidiarity* was developed in the law philosophy and social theory of the Catholic church and dates back to mid 1900. It is now also a core principle under which the European Union and its constituting member states operate.

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

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

Historically, the concept of *subsidiarity* was developed in the law philosophy and social theory of the Catholic church and dates back to mid 1900. It is now also a core principle under which the European Union and its constituting member states operate.

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**

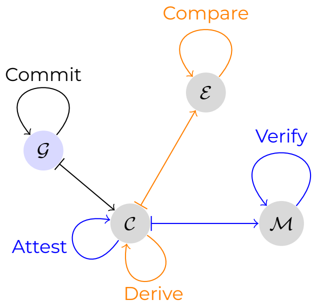
1. Tie age restriction to the **ability to pay** (not to ID's)
2. maintain **anonymity of buyers**
3. maintain **unlinkability of transactions**
4. align with **principle of subsidiarity**
5. be **practical and efficient**

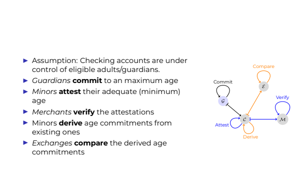
1. Binding the age-restriction to the *ability* to pay also allows to make exceptions: For instance, a guardian might allow its 14 year old child a limited amount of money with a higher maximum age, or no age-restriction attached to it. This is also an example for the application of the principle of subsidiarity.
2. In times of Blockchains it seems to be necessary to insist on the *practicality and efficiency* of such schemes.

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





1. This scheme is designed independent of any particular payment service protocol.
2. Note the difference between the *maximum* age, that a guardian commits to, and the required *minimum* (or *adequate*) age that the merchant requires. The maximum age is what the child can prove to be *at the maximum*. In a particular purchase, the required minimum age can also be lower that what the the child can proof (but not higher)
3. Of course, the scheme also needs to work for *adults*! In that case, the two participants in the scheme *guardian* and *child* are the same person. Adults simply commit themselves to the maximum age (or age group) permitted in the scheme.
4. The derive and compare operations exist to avoid that a child has to ask the guardian *again* for a new commitment after each purchase. The guardian might not be available, while the payment service usually is permanently online. Therefore, the compare step to ensure the equivalence of two age-commitments, should be performed by that system.

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 :	$(m, Q, P) \mapsto T$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \to \mathbb{T} \cup \{\bot\}$,
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 :	$(Q, Q', \beta) \mapsto b$	$\mathbb{O} \times \mathbb{O} \times \mathbb{B} \to \mathbb{Z}_2$,

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

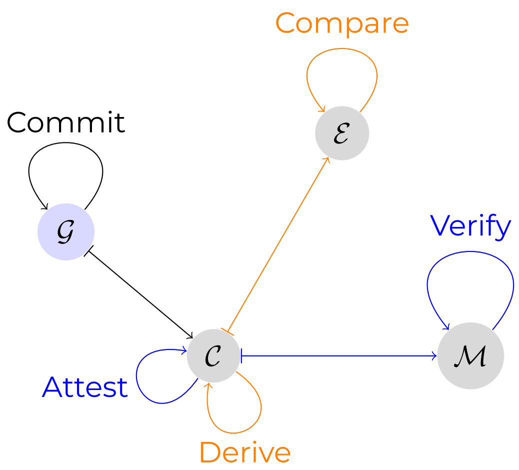
Basic and security requirements are defined later.

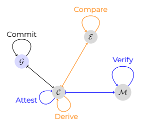
Mnemonics:
$\mathbb{O}$ = *c$\mathbb{O}$mmitments*, Q = *Q-mitment* (commitment), $\mathbb{P}$ = *$\mathbb{P}$roofs*, P = *Proof*,
$\mathbb{T}$ = *a$\mathbb{T}$testations*, T = *aTtestation*, $\mathbb{B}$ = *$\mathbb{B}$lindings*, β = *βlinding*.

Searching for functions with the following signatures
Commit : $(a, \omega) \mapsto (Q, P)$
Attest : $(m, Q, P) \mapsto T$
Verify : $(m, Q, T) \mapsto b$
Derive : $(Q, P, \omega) \mapsto (Q', P', \beta)$
Compare : $(Q, Q', \beta) \mapsto b$
with $\Omega, \mathbb{P}, \mathbb{O}, \mathbb{T}, \mathbb{B}$ sufficiently large sets.
Basic and security requirements are defined later.
Mnemonics:
$\mathbb{O}$ = cOmmitments, Q = Q-mitment (commitment), $\mathbb{P}$ = Proofs, P = Proof,
$\mathbb{T}$ = aTtestations, T = aTtestation, $\mathbb{B}$ = Blindings, β = βlinding.

1. The sets can be all considered to be sufficiently large finite subsets of $\{0, 1\}^*$.

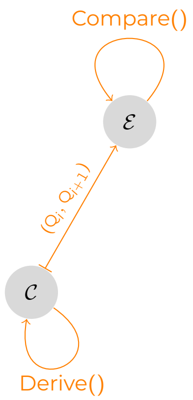
Age restriction
Naïve scheme





1. The diagram provides a sketch of the participants in the scheme, the callers and calls to the five functions and the data flow between the participants.
2. It is important to hint at the iterative nature of Derive-Compare in particular, which—in this naive version of the scheme—leads to the linkability of sequences of commitments (see next slide).

Achieving unlinkability

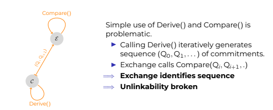


Simple use of Derive() and Compare() is problematic.

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

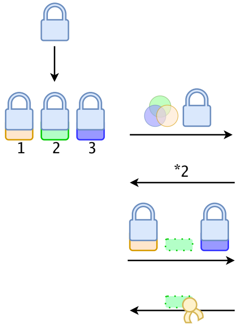
$\Longrightarrow$ **Exchange identifies sequence**

$\Longrightarrow$ **Unlinkability broken**

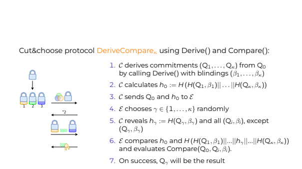


Achieving unlinkability

Cut&choose protocol DeriveCompare$_\kappa$ using Derive() and Compare():



1. $\mathcal{C}$ derives commitments $(Q_1, \ldots, Q_\kappa)$ from Q_0 by calling Derive() with blindings $(\beta_1, \ldots, \beta_\kappa)$
2. $\mathcal{C}$ calculates $h_0 := H(H(Q_1, \beta_1)||\ldots||H(Q_\kappa, \beta_\kappa))$
3. $\mathcal{C}$ sends Q_0 and h_0 to $\mathcal{E}$
4. $\mathcal{E}$ chooses $\gamma \in \{1, \ldots, \kappa\}$ 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 $H(H(Q_1, \beta_1)||...||h_\gamma||...||H(Q_\kappa, \beta_\kappa))$ and evaluates Compare(Q_0, Q_i, β_i).
7. On success, Q_γ will be the result



1. This is another example where cut&choose is used as a generic, knowledge-hiding "overlay"-protocol.
2. It leads to the same *combined* results of the calls to Derive and Compare in succession. In particular, the protocol DeriveCompare$_\kappa$, considered as function, has the signature

$$\mathbb{O} \times \mathbb{P} \times \Omega \to \mathbb{O} \times \mathbb{P} \times \mathbb{B} \times \{0, 1\}$$

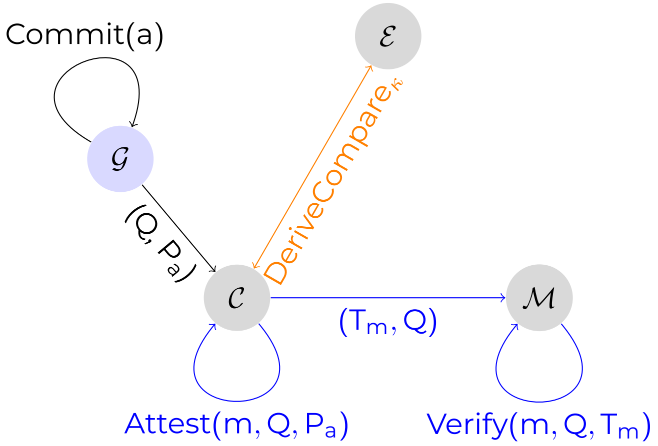
3. Scheme is similar to the *refresh* protocol in GNU Taler.

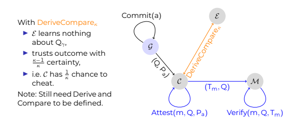
Achieving unlinkability

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.





1. The image on the right shows the refined scheme using DeriveCompare$_\kappa$ for unlinkability.
2. Consider also a variant of the scheme, where instead of sending *one* γ, $\mathcal{E}$ sends $\vec{\gamma} \in \{1, \ldots, \kappa\}^M$ (that is: one value per age-group). Then the chance for $\mathcal{C}$ to cheat would *fall* (depending on how many age groups are above the commited one and are therefore worth cheating).

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

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

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

Efficacy of attestations

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

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

etc.

1. The first one is basically that the Attest function will always be successful if asked to provide an age restriction proof for an age smaller than the commitment, and fail if the requested age is larger.
2. The second equation simply states that Verify must pass age restriction proofs generated from Attest for the correct age, and fail otherwise.
3. Equivalent intuitive requirements on the preservation of the age restriction permissions apply to Derive and Compare.

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.

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 $G^{FA}_{\mathcal{A}}(\lambda)$—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 $\text{Verify}(m, Q, T)$

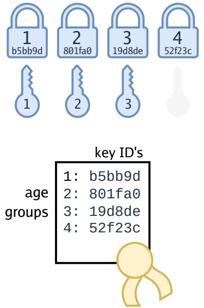
Requirement: Unforgeability of minimum age

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

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

1. This is a simplified version of the actual game from the paper [4]. In this simplified version we don't consider the Derive operation, i. e. we don't provide the adversary with additional commitments and proofs coming from a calls to Derive (instead of Commit).

Solution: Instantiation with ECDSA



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

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

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

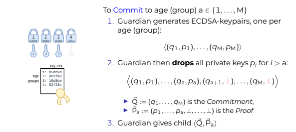
2. Guardian then **drops** all private keys p_i for $i > a$:

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

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

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

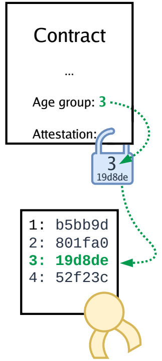
NGI TALER



1. The scheme requires M keys for M age groups.
2. Thus, some key costs are linear in the number of age groups.
3. Fortunately, below 6 and above 21 usually nobody cares, so M is small.
4. In principle, the keys could represent arbitrary capabilities of programmable money; using them for anything but age-restrictions may raise ethical concerns. GNU Taler only supports the use-case for age-restrictions.

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 $m \leq a$:

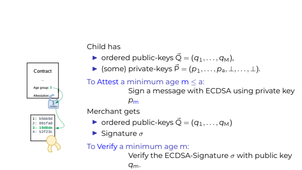
Sign a message with ECDSA using private key p_m

Merchant gets

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

To Verify a minimum age m:

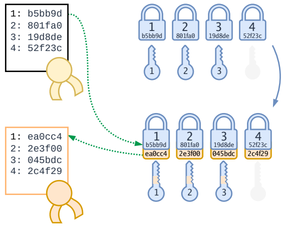
Verify the ECDSA-Signature σ with public key q_m.



1. Performance primarily matters for the user during payment.
2. All other operations usually happen in the background.
3. Thus, it is good that attestation and verification are cheap ($O(1)$) and independent of M.

Instantiation with ECDSA

Definitions of Derive and Compare



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

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

Choose random $\beta \in \mathbb{Z}_g$ and calculate

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

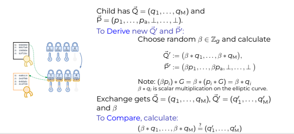
Note: $(\beta p_i) * G = \beta * (p_i * G) = \beta * q_i$

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

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

To Compare, calculate:

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



1. The cryptography used for blinding is inspired by [6].

Instantiation with ECDSA

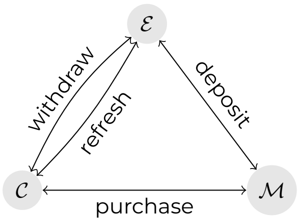
Functions (Commit, Attest, Verify, Derive, Compare)
as defined in the instantiation with ECDSA

- meet the basic requirements,
- also meet all security requirements.
  Proofs by security reduction, details are in the paper [4].

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 [4].

Integration with GNU Taler

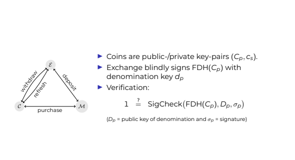
Key operations in the original system



- 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)



The diagram on the left now is a sketch of the main protocols of GNU Taler. The participants in our scheme and in GNU Taler are almost the same: The guardian is missing in GNU Taler.

Integration with GNU Taler

Binding age restriction to coins

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

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

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

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

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

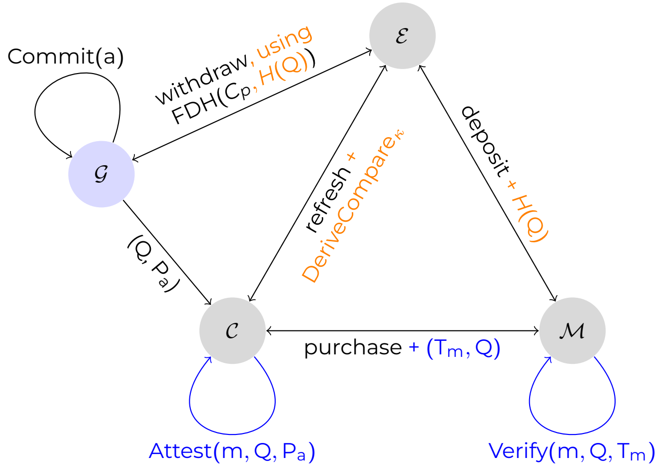
$$\text{FDH}(C_p, H(\text{Q}))$$

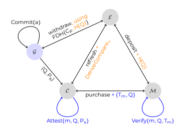
Verfication of a coin now requires $H(\text{Q})$, too:

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

Integration with GNU Taler

Integrated schemes





1. In the diagram we do not mention the actual coin material. For example, the guardian not only gives the age-commitment and -proof to the child, but also the corresponding coin's private key to which the commitment is bound to.
2. In an actual implementation of the subsidiary flow of the age-restriction scheme, one might choose to place the Commit and withdraw step actually onto the wallet of the *child*, and instruct parents to make sure that the child has set appropriate age-restrictions prior to filling the reserve.
3. Note that any two coins of the same denomination and with age restriction set are still indistinguishable.

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.

Paper also formally defines another signature scheme: Edx25519.

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

Current implementation of age restriction in GNU Taler uses Edx25519.

Discussion

- 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.

- 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.

1. It is worth noting that the scheme would not work as a pure age-verification scheme, *independent* of any payment: Without the financial punishment for cheating, the child can try DeriveCompare$_\kappa$ often enough until it succeeds in gets the maximal possible age commitment.

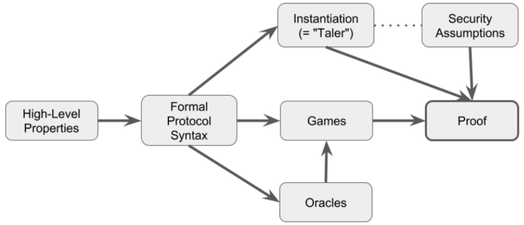
What are the future plans for GNU Taler?

What are the future plans for GNU Taler?

Summary

GNU Taler:

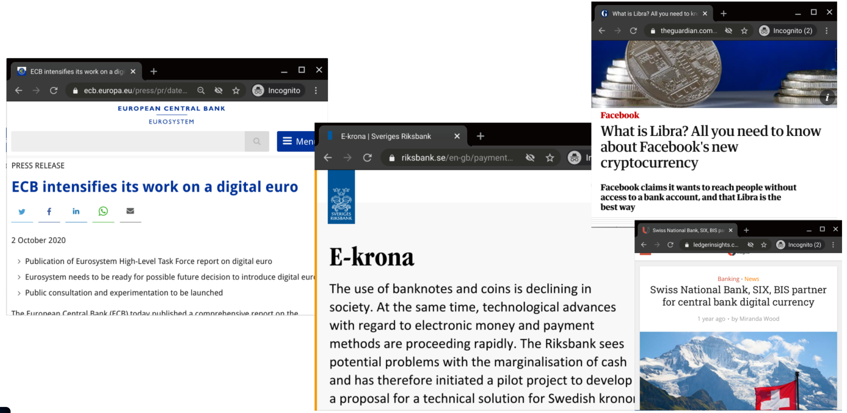
- Gives change, can provide refunds
- Integrates nicely with HTTP, handles network failures
- Has high performance
- Is Free Software
- Includes formal security proofs

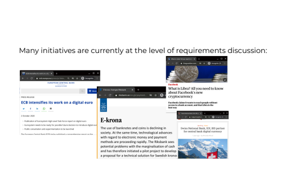


GNU Taler:
- Gives change, can provide refunds
- Integrates nicely with HTTP, handles network failures
- Has high performance
- Is Free Software
- Includes formal security proofs

CBDC initiatives and GNU Taler

Many initiatives are currently at the level of requirements discussion:





1. CBDC = Central Bank Digital Currency, so digital payment systems operated by the central bank and where the money is a liability of the central bank.
2. Taler can serve as the foundation for a *bearer-based retail* CBDC.
3. Taler replicates physical cash rather than bank deposits

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!

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!

1. ⇒ monetary policy remains under CB control
2. ⇒ no competition for commercial banking (S&L) and
   ⇒ CB does not have to manage KYC, customer support
3. ⇒ protects against bank runs and hoarding
4. ⇒ additional insights into economy and new policy options
5. ⇒ exit strategy and handles catastrophic security incidents
6. ⇒ **CB cannot be forced to facilitate mass-surveillance**

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.

- 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.

1. This list naturally changes frequently.
2. The key message at this time is that Taler is currently operated at a small scale, but we expect to grow it much bigger soon.
3. Help with translations is always welcome.

Open issues

- Address remaining scalability challenges (multiple topics)
- Porting to more platforms (Web shops, iOS, embedded, ...)
- 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/`)

- Address remaining scalability challenges (multiple topics)
- Porting to more platforms (Web shops, iOS, embedded, ...)
- 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/)

1. GNU Taler already performs well, but there are many, many scenarios to test and further optimize. So plenty of work to go around!
2. Similarly, payments have many applications, so integrating GNU Taler into any business process that involves a customer payment is literally work that will never end. But, it should be simple if you actually understand the business process.
3. Usability is another critical topic, and we hope to address the needs of more than just the 99% of commercially viable users.

Visions

- Be paid to read advertising, starting with spam
- Give welfare without intermediaries taking huge cuts
- Forster regional trade via regional currencies
- Eliminate corruption by making all income visible
- Stop the mining by making crypto-currencies useless for anything but crime

- Be paid to read advertising, starting with spam
- Give welfare without intermediaries taking huge cuts
- Forster regional trade via regional currencies
- Eliminate corruption by making all income visible
- Stop the mining by making crypto-currencies useless for anything but crime

1. These are some of the more speculative results that we hope to eventually achieve using GNU Taler.
2. Do you have any interesting ideas of what one could also accomplish with this system?

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.

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.

References II

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.

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.

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.

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.

References III

- 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.
- Phillip Rogaway.
  The moral character of cryptographic work.
  Austin, TX, August 2016. USENIX Association.

- 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.
- Phillip Rogaway.
  The moral character of cryptographic work.
  Austin, TX, August 2016. USENIX Association.

References IV

Martin Schanzenbach, Christian Grothoff, and Bernd Fix.
The GNU Name System.
RFC 9498, November 2023.

Martin Schanzenbach, Christian Grothoff, and Bernd Fix.
The GNU Name System.
RFC 9498, November 2023.

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.