Non-boring cryptography

Christian Grothoff

Berner Fachhochschule

7.6.2019

Learning Objectives

- Blind signatures and applications
- Homomorphic encryption
- Secure multiparty computation (SMC) theory
- Common SMC adversary models
- Specialized protocols for SMC problems

Reminder: RSA

Pick p, q prime and e such that

$$GCD((p-1)(q-1), e) = 1 \tag{1}$$

- Define $n = pq$,
- compute d such that $ed \equiv 1 \mod (p-1)(q-1)$.
- Let $s := m^d \mod n$.
- Then $m \equiv s^e \mod n$.

RSA Summary

- Public key: n, e
- Private key: $d \equiv e^{-1} \mod \phi(n)$ where $\phi(n) = (p-1) \cdot (q-1)$
- Encryption: $c \equiv m^e \mod n$
- Decryption: $m \equiv c^d \mod n$
- Signing: $s \equiv m^d \mod n$
- Verifying: $m \equiv s^e \mod n$?

Low Encryption Exponent Attack

- e is known
- M maybe small
- $C = M^e < n$?
- If so, can compute $M = \sqrt[e]{C}$

$\Rightarrow$ Small e can be bad!

Padding and RSA Symmetry

- Padding can be used to avoid low exponent issues (and issues with $m = 0$ or $m = 1$)
- Randomized padding defeats chosen plaintext attacks
- Padding breaks RSA symmetry:

$$D_{A_{priv}}(D_{B_{priv}}(E_{A_{pub}}(E_{B_{pub}}(M)))) \neq M \tag{2}$$

- PKCS#1 / RFC 3447 define a padding standard

Blind signatures with RSA

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

Blind signatures with RSA

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

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

Blind signatures with RSA

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

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

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

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

Surveilance concerns

- Everybody knows about Internet surveilance.
- But is it **that** bad?

Surveilance concerns

- Everybody knows about Internet surveilance.
- But is it **that** bad?
  - You can choose when and where to use the Internet
  - You can anonymously access the Web using Tor
  - You can find open access points that do not require authentication
  - IP packets do not include your precise location or name
  - ISPs typically store this meta data for days, weeks or months

Where is it worse?

This was a question posed to RAND researchers in 1971:

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

Where is it worse?

This was a question posed to RAND researchers in 1971:

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

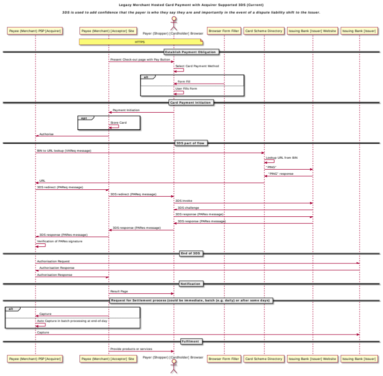
What is worse:

- 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 at least 6 years

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 sovereingity is in danger

支付宝™
Alipay.com

PayPal™

Pay

Pay with Amazon

SAMSUNG
pay

pay

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

Do you want to live under total surveillance?

Banking, Surveillance and Physical Security

`https://www.youtube.com/watch?v=GyJZViNf2Vk` (4'2019)

Break

Bitcoin

Payment system using a block chain:

- Public keys identify accounts, private keys used to send money from the account into other accounts.
- Set of internally consistent transactions form block
- Each block includes a transaction creating fresh coins and transferring applicable fees to block creator
- Difficulty adjusts to mining power to mine a block in ≈ 10 minutes
- Amount of coins created per block is exponentially decreasing

Mining

Mining requires:

- Learning pending transactions from peers
- Selecting a subset of of transactions which is valid (no double spending) by computing current account balances against the entire history
- Finding a hash collision (with adaptive difficulty)
- Propagating the new block to other miners

Usually specialized systems are used for finding hash collisions.

The CAP Theorem

We like three properties in distributed systems:

- Consistency: everybody agrees on the state
- Availability: the system is always available
- Partition tolerance: the system handles component (host, link) failure, including network partitions

CAP Theorem is an **impossibility proof**: you can only have 2/3.

CAP & Bitcoin

Bitcoin is inconsistent:

- Conflicting blocks can be mined at the same time
- This can happen by accident, or on purpose!
- Coins could be spent twice, once on each fork of the chain!
- Longest chain is considered “valid”
- Original paper suggests to consider transaction confirmed only after at least 6 blocks past the transaction.

$\Rightarrow$ Bitcoin is not consistent.
$\Rightarrow$ Competitively long alternative chains void durability even after 6 blocks!

Bitcoin performance

- Privacy: all transactions happen in the clear in public view
- Latency: transactions take 1h to kind-of be confirmed
- Storage: grows linearly forever, no garbage collection
- Power: mining consumes more than the entire state of Denmark today
- Rate: Network handles at most about 7 transactions per second
- Accountability: use of public keys as addresses enables criminal use

$\Rightarrow$ Bitcoin fever lasting for years. Why?

Altcoins

- Dogecoin: same as Bitcoin, just named after a dog meme (an idea that is obviously worth billions!)
- Zcash: uses ZKSNARKs[1] to hide transactions (criminal activity on Bitcoin was too low)
- Ethereum: run Turing-complete virtual machine logic in the blockchain to enable “smart” contracts and arbitrary applications, not just payments (is “Accelerando” an utopia or dystopia?)

Experimental designs promising to drastically improve performance (Bolt, Lightning) have so far failed to deliver.

[1] $\approx$ 1-15 minutes CPU time to create new transaction needed!

Case study: Private payments

"A company is developing new software for private payments. This will enable its customers to transact with "complete" privacy (like cash). The solution does not include backdoors, and thus the company cannot block payments to support trade embargos or anti money laundering efforts."

- Discuss virtues and vices affected.
- Does it make a difference if the software is Free Software developed by a community instead of proprietary software from a company?
- Suppose the company added a feature to provide income transparency where the state gets to see who receives funds (but not who made payments). Does this change your assessment?

Break

GNU Taler

Digital cash, made **socially responsible**.

⟨Taler⟩

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

What is Taler?

Taler is an electronic instant payment system.

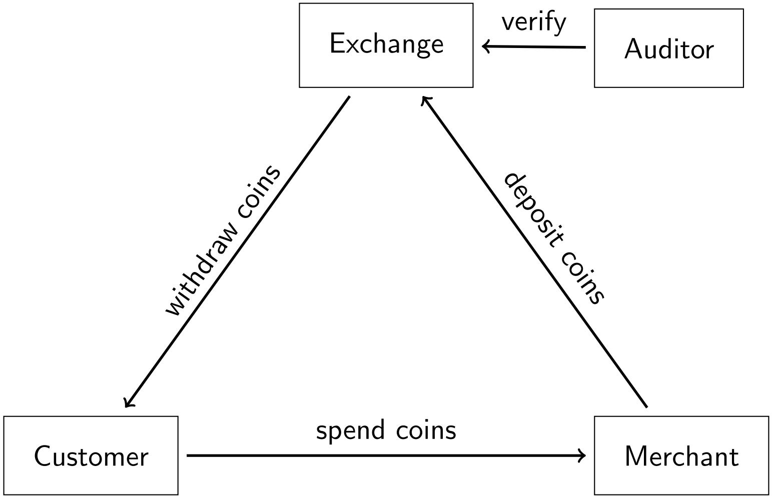
- Uses electronic coins stored in **wallets** on customer's device
- Like **cash**
- Pay in **existing currencies** (i.e. EUR, USD, BTC),
  or use it to create new **regional currencies**

Design goals for the GNU Taler Payment System

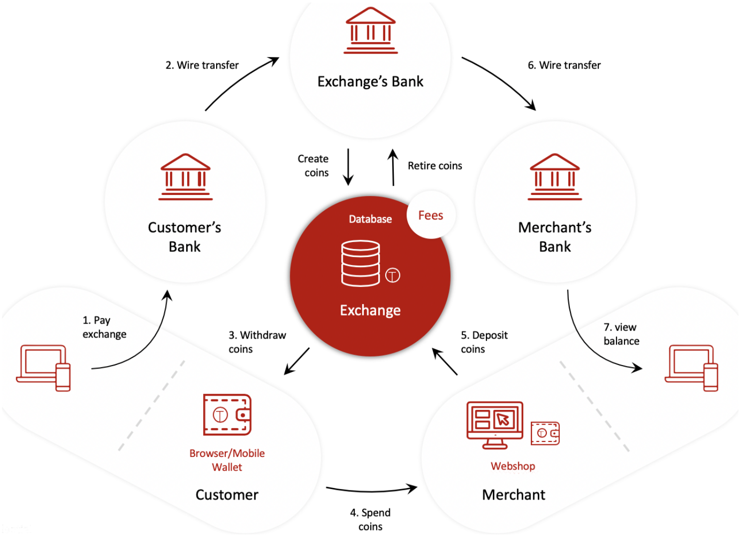
GNU Taler must ...

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

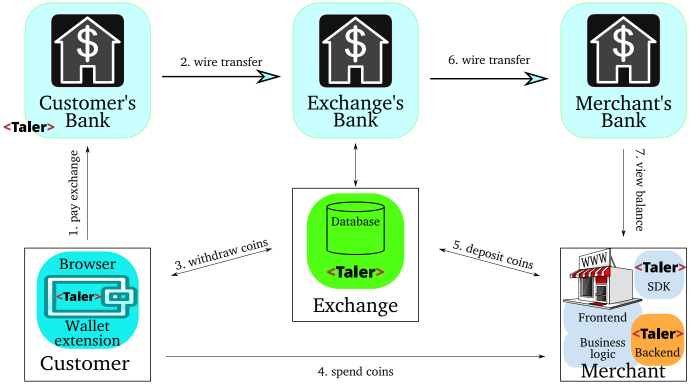
Taler Overview



Architecture of Taler



Architecture of Taler



⇒ Convenient, taxable, privacy-enhancing, & resource friendly!

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.

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.

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

How does it work?

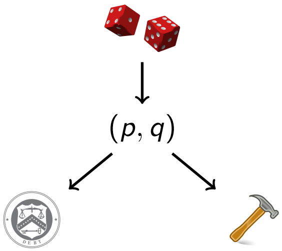
We use a few ancient constructions:

- Cryptographic hash function (1989)
- Blind signature (1983)
- Schnorr signature (1989)
- Diffie-Hellman key exchange (1976)
- Cut-and-choose zero-knowledge proof (1985)

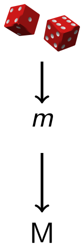
But of course we use modern instantiations.

Exchange setup: Create a denomination key (RSA)

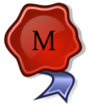
1. Pick random primes p, q.
2. Compute $n := pq$, $\phi(n) = (p-1)(q-1)$
3. Pick small $e < \phi(n)$ such that $d := e^{-1} \mod \phi(n)$ exists.
4. Publish public key (e, n).



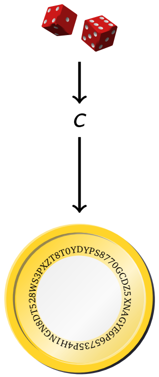
Merchant: Create a signing key (EdDSA)



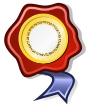
- pick random $m \mod o$ as private key
- $M = mG$ public key

Capability: $m \Rightarrow$ 

Customer: Create a planchet (EdDSA)

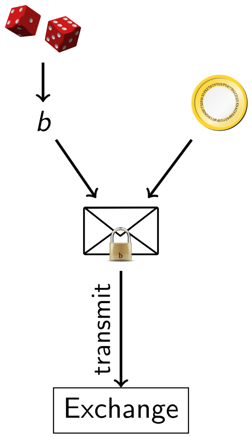


- Pick random $c \mod o$ private key
- $C = cG$ public key

Capability: $c \Rightarrow$ 

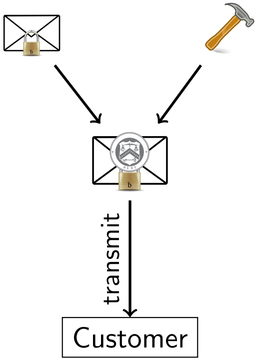
Customer: Blind planchet (RSA)

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



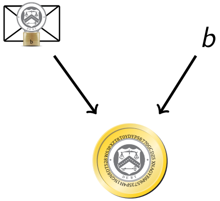
Exchange: Blind sign (RSA)

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

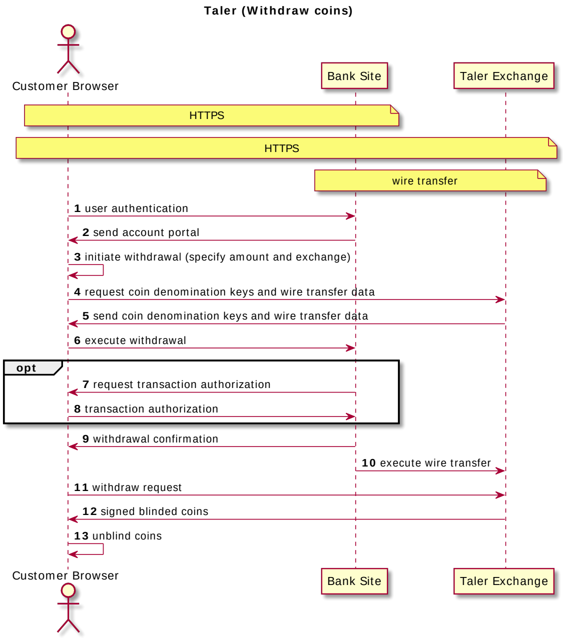


Customer: Unblind coin (RSA)

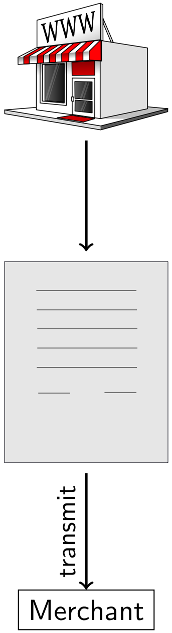
1. Receive s'.
2. Compute $s := s'b^{-1} \mod n$



Withdrawing coins on the Web

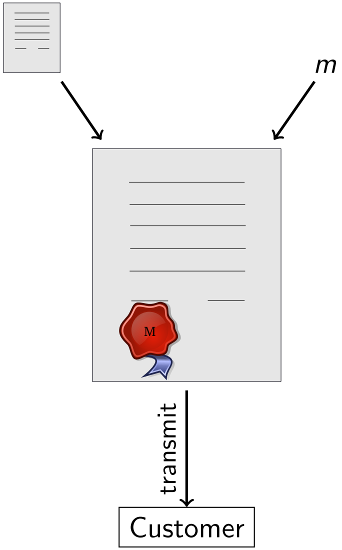


Customer: Build shopping cart



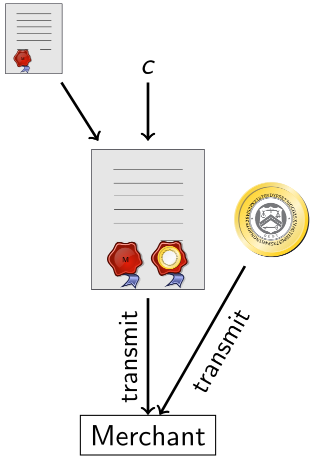
Merchant: Propose contract (EdDSA)

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



Customer: Spend coin (EdDSA)

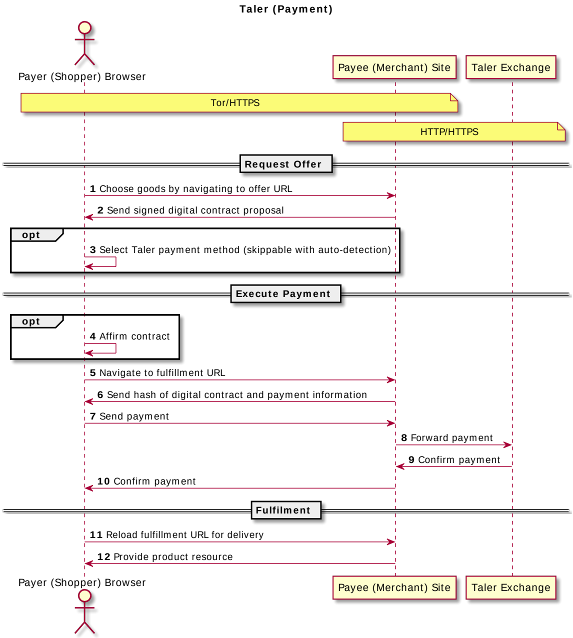
1. Receive proposal D, $EdDSA_m(D)$.
2. Send s, C, $EdDSA_c(D)$



Merchant and Exchange: Verify coin (RSA)

$$s^e \overset{?}{\equiv} m \mod n$$

Payment processing with Taler



Taler (Payment)
Payer (Shopper) Browser
Payee (Merchant) Site
Taler Exchange
Tor/HTTPS
HTTP/HTTPS
Request Offer
1 Choose goods by navigating to offer URL
2 Send signed digital contract proposal
opt
3 Select Taler payment method (skippable with auto-detection)
Execute Payment
opt
4 Affirm contract
5 Navigate to fulfillment URL
6 Send hash of digital contract and payment information
7 Send payment
8 Forward payment
9 Confirm payment
10 Confirm payment
Fulfilment
11 Reload fulfillment URL for delivery
12 Provide product resource
Payer (Shopper) Browser
Payee (Merchant) Site
Taler Exchange

Warranting deposit safety

Exchange has *another* online signing key $W = wG$:

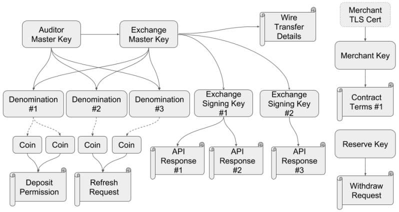
Sends E, $EdDSA_w(M, H(D), FDH(C))$ to the merchant.

This signature means that M was the *first* to deposit C and that the exchange thus must pay M.

Without this, an evil exchange could renege on the deposit confirmation and claim double-spending if a coin were deposited twice, and then not pay either merchant!

Online keys

- The exchange needs d and w to be available for online signing.
- The corresponding public keys W and (e, n) are certified using Taler's public key infrastructure (which uses offline-only keys).

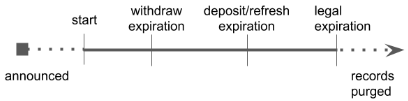


What happens if those private keys are compromised?

Denomination key (e, n) compromise

- An attacker who learns d can sign an arbitrary number of illicit coins into existence and deposit them.
- Auditor and exchange can detect this once the total number of deposits (illicit and legitimate) exceeds the number of legitimate coins the exchange created.
- At this point, (e, n) is *revoked*. Users of *unspent* legitimate coins reveal b from their withdrawal operation and obtain a *refund*.
- The financial loss of the exchange is *bounded* by the number of legitimate coins signed with d.

$\Rightarrow$ Taler frequently rotates denomination signing keys and deletes d after the signing period of the respective key expires.



Online signing key W compromise

- An attacker who learns w can sign deposit confirmations.
- Attacker sets up two (or more) merchants and customer(s) which double-spend legitimate coins at both merchants.
- The merchants only deposit each coin once at the exchange and get paid once.
- The attacker then uses w to fake deposit confirmations for the double-spent transactions.
- The attacker uses the faked deposit confirmations to complain to the auditor that the exchange did not honor the (faked) deposit confirmations.

The auditor can then detect the double-spending, but cannot tell who is to blame, and (likely) would presume an evil exchange, forcing it to pay both merchants.

Break

Giving change

It would be inefficient to pay EUR 100 with 1 cent coins!

- Denomination key represents value of a coin.
- Exchange may offer various denominations for coins.
- Wallet may not have exact change!
- Usability requires ability to pay given sufficient total funds.

Giving change

It would be inefficient to pay EUR 100 with 1 cent coins!

- Denomination key represents value of a coin.
- Exchange may offer various denominations for coins.
- Wallet may not have exact change!
- Usability requires ability to pay given sufficient total funds.

Key goals:

- maintain unlinkability
- maintain taxability of transactions

Giving change

It would be inefficient to pay EUR 100 with 1 cent coins!

- Denomination key represents value of a coin.
- Exchange may offer various denominations for coins.
- Wallet may not have exact change!
- Usability requires ability to pay given sufficient total funds.

Key goals:

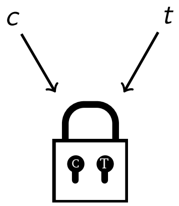
- maintain unlinkability
- maintain taxability of transactions

Method:

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

Diffie-Hellman (ECDH)

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

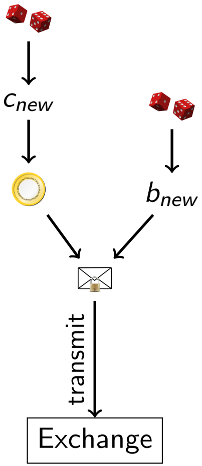


Strawman solution

Given partially spent private coin key c_{old}:

1. Pick random $c_{new} \mod o$ private key
2. $C_{new} = c_{new} G$ public key
3. Pick random b_{new}
4. Compute $f_{new} := FDH(C_{new})$, $m < n$.
5. Transmit $f'_{new} := f_{new} b^e_{new} \mod n$

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

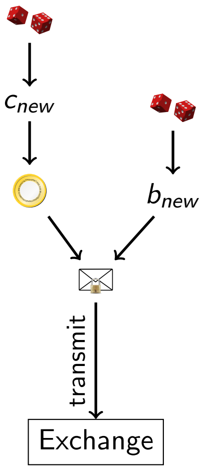


Strawman solution

Given partially spent private coin key c_{old}:

1. Pick random $c_{new} \mod o$ private key
2. $C_{new} = c_{new} G$ public key
3. Pick random b_{new}
4. Compute $f_{new} := FDH(C_{new})$, $m < n$.
5. Transmit $f'_{new} := f_{new} b_{new}^e \mod n$

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

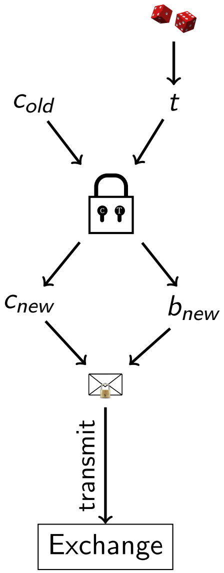


Problem: Owner of c_{new} may differ from owner of c_{old}!

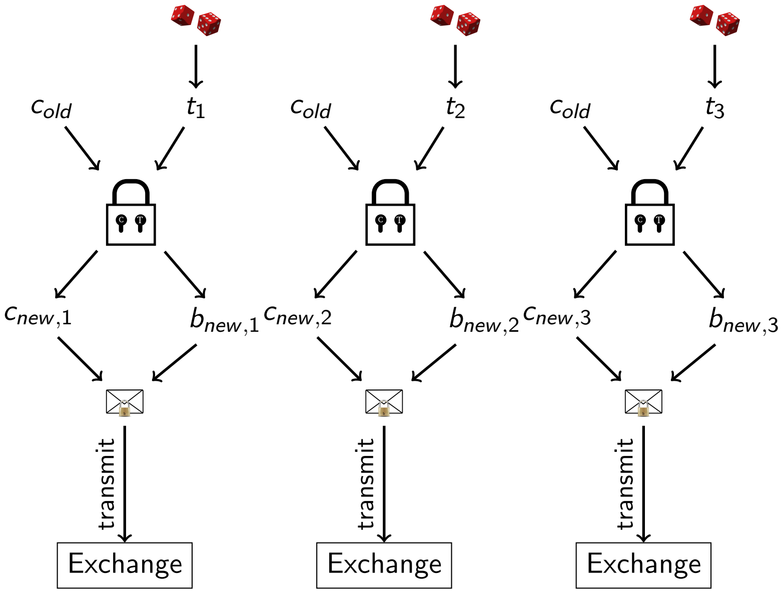
Customer: Transfer key setup (ECDH)

Given partially spent private coin key c_{old}:

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



Cut-and-Choose



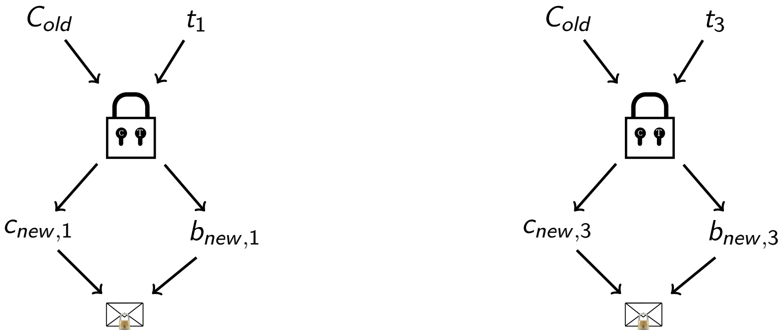
Exchange: Choose!

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

Customer: Reveal

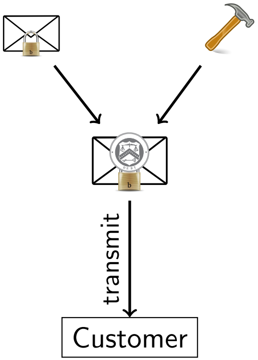
1. If $\gamma = 1$, send t_2, t_3 to exchange
2. If $\gamma = 2$, send t_1, t_3 to exchange
3. If $\gamma = 3$, send t_1, t_2 to exchange

Exchange: Verify ($\gamma = 2$)



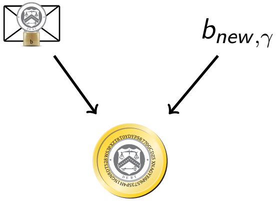
Exchange: Blind sign change (RSA)

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



Customer: Unblind change (RSA)

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

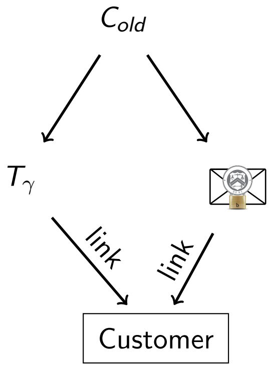


Exchange: Allow linking change

Given C_{old}

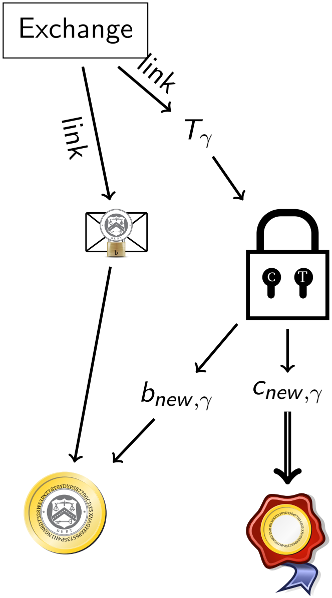
return T_γ and

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



Customer: Link (threat!)

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



Refresh protocol summary

- Customer asks exchange to convert old coin to new coin
- Protocol ensures new coins can be recovered from old coin

$\Rightarrow$ New coins are owned by the same entity!

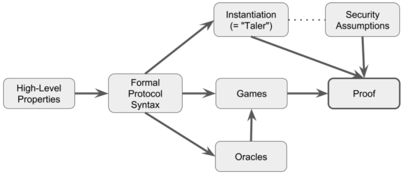
Thus, the refresh protocol allows:

- 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.

Summary

- We can design protocols that fail *soft*.
- GNU Taler's design limits financial damage even in the case private keys are compromised.
- GNU Taler does:
  - Gives change, can provide refunds
  - Integrates nicely with HTTP, handles network failures
  - High performance
  - Free Software
  - Formal security proofs



Next Steps

- In discussions with various banks (commercial and central banks)
- In discussions with investors
- Implementation still needs:
  - Escrowed wallet backup and synchronization solution
  - Finish integration with existing banking system (EBICS, FinTS)
  - Code security audit
  - Improved design and usability
  - Internationalization
  - Porting to more platforms (Web shops, Android, POS)
  - Launch partners
- Regulatory approval (withdraw and deposit limits, independent auditor, KYC/AML process validation)

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

Break

Secure Multiparty Computation (SMC)

- Alice und Bob haben private Daten a_i and b_i.
- Alice und Bob führen ein Protokoll aus und berechnen gemeinsam $f(a_i, b_i)$.
- Nur einer von beiden lernt das Ergebnis (i.d.R.)

Adversary model

Honest but curious

Homomorphic Encryption

$$E(x_1 \oplus x_2) = E(x_1) \otimes E(x_2) \tag{3}$$

Multiplicative Homomorphism: RSA & ElGamal

- Unpadded RSA (multiplicative):

$$E(x_1) \cdot E(x_2) = x_1^e x_2^e = E(x_1 \cdot x_2) \tag{4}$$

- ElGamal:

$$\begin{aligned} E(x_1) \cdot E(x_2) &= (g^{r_1}, x_1 \cdot h^{r_1})(g^{r_2}, x_2 \cdot h^{r_2}) && (5) \\ &= (g^{r_1+r_2}), (x_1 \cdot x_2) h^{r_1+r_2}) && (6) \\ &= E(m_1 \cdot m_2) && (7) \end{aligned}$$

Additive Homomorphism: Paillier

$$\mathsf{E}_K(m) := g^m \cdot r^n \mod n^2, \tag{8}$$

$$\mathsf{D}_K(c) := \frac{(c^\lambda \mod n^2) - 1}{n} \cdot \mu \mod n \tag{9}$$

where the public key $K = (n, g)$, m is the plaintext, c the ciphertext, n the product of $p, q \in \mathbb{P}$ of equal length, and $g \in \mathbb{Z}^*_{n^2}$. In Paillier, the private key is (λ, μ), which is computed from p and q as follows:

$$\lambda := \mathrm{lcm}(p-1, q-1), \tag{10}$$

$$\mu := \left(\frac{(g^\lambda \mod n^2) - 1}{n}\right)^{-1} \mod n. \tag{11}$$

Paillier offers additive homomorphic public-key encryption, that is:

$$\mathsf{E}_K(a) \otimes \mathsf{E}_K(b) \equiv \mathsf{E}_K(a + b) \tag{12}$$

for any public key K.

Fully homomorphic encryption

Additive:

$$E(A) \oplus E(B) = E(A + B) \tag{13}$$

and multiplicative:

$$E(A) \otimes E(B) = E(A \cdot B) \tag{14}$$

Known cryptosystems: Brakerski-Gentry-Vaikuntanathan (BGV), NTRU, Gentry-Sahai-Waters (GSW).

Break

Example: Secure Scalar Product

- Original idea by Ioannids et al. in 2002 (use: $(a-b)^2 = a^2 - 2ab + b^2$)
- Refined by Amirbekyan et al. in 2007 (corrected math)
- Now providing protocol with practical extensions (negative numbers, small numbers, set intersection).

Preliminaries

- Alice has public key A and input map $m_A : M_A \to \mathbb{Z}$.
- Bob has public key B and input map $m_B : M_B \to \mathbb{Z}$.
- We want to calculate

$$\sum_{i \in M_A \cap M_B} m_A(i) m_B(i) \tag{15}$$

- We first calculate $M = M_A \cap M_B$.
- Define $a_i := m_A(i)$ and $b_i := m_B(i)$ for $i \in M$.
- Let s denote a shared static offset.

Network Protocol

- Alice transmits $\mathsf{E}_A(s + a_i)$ for $i \in M$ to Bob.
- Bob creates two random permutations π and π' over the elements in M, and a random vector r_i for $i \in M$ and sends

$$\begin{aligned}
R :&= \mathsf{E}_A(s + a_{\pi(i)}) \otimes \mathsf{E}_A(s - r_{\pi(i)} - b_{\pi(i)}) && (16)\\
&= \mathsf{E}_A(2 \cdot s + a_{\pi(i)} - r_{\pi(i)} - b_{\pi(i)}), && (17)\\
R' :&= \mathsf{E}_A(s + a_{\pi'(i)}) \otimes \mathsf{E}_A(s - r_{\pi'(i)}) && (18)\\
&= \mathsf{E}_A(2 \cdot s + a_{\pi'(i)} - r_{\pi'(i)}), && (19)\\
S :&= \sum (r_i + b_i)^2, && (20)\\
S' :&= \sum r_i^2 && (21)
\end{aligned}$$

Decryption (1/3)

Alice decrypts R and R' and computes for $i \in M$:

$$a_{\pi(i)} - b_{\pi(i)} - r_{\pi(i)} = \mathrm{D}_A\left(R\right) - 2 \cdot s, \tag{22}$$

$$a_{\pi'(i)} - r_{\pi'(i)} = \mathrm{D}_A\left(R'\right) - 2 \cdot s, \tag{23}$$

which is used to calculate

$$T := \sum_{i \in M} a_i^2 \tag{24}$$

$$U := -\sum_{i \in M} (a_{\pi(i)} - b_{\pi(i)} - r_{\pi(i)})^2 \tag{25}$$

$$U' := -\sum_{i \in M} (a_{\pi'(i)} - r_{\pi'(i)})^2 \tag{26}$$

Decryption (2/3)

She then computes

$$\begin{aligned}
P &:= S + T + U \\
&= \sum_{i\in M}(b_i + r_i)^2 + \sum_{i\in M} a_i^2 + \left(-\sum_{i\in M}(a_i - b_i - r_i)^2\right) \\
&= \sum_{i\in M}\left((b_i + r_i)^2 + a_i^2 - (a_i - b_i - r_i)^2\right) \\
&= 2 \cdot \sum_{i\in M} a_i(b_i + r_i). \\
P' &:= S' + T + U' \\
&= \sum_{i\in M} r_i^2 + \sum_{i\in M} a_i^2 + \left(-\sum_{i\in M}(a_i - r_i)^2\right) \\
&= \sum_{i\in M}\left(r_i^2 + a_i^2 - (a_i - r_i)^2\right) = 2 \cdot \sum_{i\in M} a_i r_i.
\end{aligned}$$

Decryption (3/3)

Finally, Alice computes the scalar product using:

$$\frac{P - P'}{2} = \sum_{i \in M} a_i(b_i + r_i) - \sum_{i \in M} a_i r_i = \sum_{i \in M} a_i b_i. \qquad (27)$$

Computing Discrete Logarithms

Who said calculating DLOG was hard?

ECC Version[2]

Alice's public key ist $A = g^a$, ihr private key ist a. Alices schickt an Bob $(g_i, h_i) = (g^{r_i}, g^{r_i a + a_i})$ mit zufälligen Werten r_i für $i \in M$. Bob antwortet mit

$$\left(\prod_{i \in M} g_i^{b_i}, \prod_{i \in M} h_i^{b_i}\right) = \left(\prod_{i \in M} g_i^{b_i}, (\prod_{i \in M} g_i^{b_i})^a g^{\sum_{i \in M} a_i b_i}\right)$$

Alice kann dann berechnen

$$\left(\prod_{i \in M} g_i^{b_i}\right)^{-a} \cdot \left(\prod_{i \in M} g_i^{b_i}\right)^{a} \cdot g^{\sum_{i \in M} a_i b_i} = g^{\sum_{i \in M} a_i b_i}.$$

Falls $\sum_{i \in M} a_i b_i$ ausreichend klein ist, kann Alice dann das Skalarprodukt durch Lösung des DLP bestimmen.

[2] Joint work with Tanja Lange

Performance Evaluation

Length	RSA-2048	ECC-2^{20}	ECC-2^{28}
25	14 s	2 s	29 s
50	21 s	2 s	29 s
100	39 s	2 s	29 s
200	77 s	3 s	30 s
400	149 s	OOR	31 s
800	304 s	OOR	33 s
800	3846 kb	OOR	70 kb

The pre-calculation of ECC-2^{28} is $\times 16$ more expensive than for ECC-2^{20} as the table is set to have size $\sqrt{n}$.

Exercise

Implement function to calculate DLOG.