Surviving Private Key Compromise in Electronic Payment Systems

GNU

〈Taler〉

taler.net
IRC＃taler
(on freenode)
twitter@taler
mail@taler.net

Florian Dold & Christian Grothoff
{dold,grothoff}@taler.net
Prelude: draft-dold-payto

payto://

See also:
https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml
<table>
<thead>
<tr>
<th>Empfängerbank / Kreditinstitut / Rechnunggeber</th>
<th>Empfänger / Konto / Rechnungsnummer</th>
<th>Vermerk / Konto / Rechnungsnummer</th>
<th>Versichertes Girokonto</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seldwyla Bank 8001 Zürich</td>
<td>N. Muster AG Versandhaus Industriestrasse 88 8000 Zürich</td>
<td>01 - 145-6 OF 2830.50 21 67000 00075 20033 45590 00126</td>
<td>Rutschnann Pia Marktgasse 28 9400 Rorschach</td>
</tr>
</tbody>
</table>

**SEPA-Überweisung/Zahlungseingang**

**VR Bank Schwäbisch Hall eG**

**DE12500004149876543720**

**BIC des Kreditinstitutes/Unternehmens**

**GEBERDESBA**

**Für Überweisungen in Deutschland und in andere EDW-Nutzer-Staaten in Europa**

**Angebote zu Zahlungseingängen**

<table>
<thead>
<tr>
<th>Bankleitzahl</th>
<th>Kontonummer</th>
</tr>
</thead>
<tbody>
<tr>
<td>3007000000</td>
<td>10000000</td>
</tr>
</tbody>
</table>

**Konto-Bestätigungsnummer**

**DE12500004149876543720**

**Datum**

**01**
payto: Uniform Identifiers for Payments and Accounts

Like mailto:, but for bank accounts instead of email accounts!

payto://<PAYMENT-METHOD>/<ACCOUNT-NR>
  ?subject=InvoiceNr42
  &amount=EUR:12.50

Default action: Open app to review and confirm payment.
Benefits of Payto

- Standardized way to represent financial resources (bank account, bitcoin wallet) and payments to them
- Useful on the client-side on the Web and for FinTech backend applications
- Payment methods (such as IBAN, ACH, Bitcoin) are registered with IANA and allow extra options
GNU Taler

Digital cash, made socially responsible.

Privacy-Preserving, Practical, Taxable, Free Software, Efficient
What is Taler?

Taler is an electronic instant payment system suitable for a CBEC.

- Uses electronic coins stored in wallets on customer’s device
- Like cash
- Pay in existing currencies (i.e. EUR, USD, BTC)
Taler Overview

- Exchange
  - Customer
    - withdraw coins
  - Auditor
    - verify
  - Merchant
    - deposit coins
  - spend coins
Architecture of Taler

⇒ Convenient, taxable, privacy-enhancing, & resource friendly!
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: \( m \Rightarrow M \)
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 \(f' := f b^e \mod n\)
Exchange: Blind sign (RSA)

1. Receive $f'$.
2. Compute $s' := f'^d \mod n$.
3. Send signature $s'$.
1. Receive $s'$.
2. Compute $s := s' b^{-1} \mod n$
Customer: Build shopping cart
Merchant: Propose contract (EdDSA)

1. Complete proposal $D$.
2. Send $D$, $EdDSA_m(D)$
Customer: Spend coin (EdDSA)

1. Receive proposal $D$, $EdDSA_m(D)$.
2. Send $s$, $C$, $EdDSA_c(D)$
Merchant and Exchange: Verify coin (RSA)

\[ s^e \equiv FDH(C) \mod n \]
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 \(d\).

⇒ 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.
Detecting online signing key $W$ compromise

- Merchants are required to *probabilistically* report signed deposit confirmations to the auditor.
- Auditor can thus detect exchanges not reporting signed deposit confirmations.

$\Rightarrow$ Exchange can rekey if illicit key use is detected, then only has to honor deposit confirmations it already provided to the auditor and those without proof of double-spending and those merchants reported to the auditor.

$\Rightarrow$ Merchants that do not participate in reporting to the auditor risk their deposit permissions being voided in cases of an exchange’s private key being compromised.
Summary and further reading

- 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 more:
  - Gives change, can provide refunds
  - Integrates nicely with HTTP, handles network failures
  - High performance
  - Free Software
  - Formal security proofs

More information at https://taler.net/.
How to support?

- GNU, TUM, INRIA and BFH are *not* banks.
- We created Taler Systems SA for commercial support and development of GNU Taler.
- We are in discussions with central banks, commercial banks, suppliers, merchants and various Free Software projects to get GNU Taler into operation.
- More banking partners and venture capital would be welcome.

Talk to us!
Do you have any questions?

References:


