Taler: Usable, privacy-preserving payments for the Web

ABSTRACT

Taler is a new electronic online payment system which provides anonymity for customers and, due to this design choice, also offers significantly better usability. This paper describes the interaction processes of online payment systems, and analytically compares their usability for both customers and merchants. We then focus on the resulting assurances that Taler provides, as—particularly for payment systems—usability and security are entertwined. Web payment systems must also face the reality of constraints imposed by modern Web browser security architecture, so the analysis includes considerations of how Web payment systems exploit the security infrastructure provided by the modern Web.

1. INTRODUCTION

The future Internet needs a secure, usable and privacy-preserving micropayment system that is not backed by a "crypto currency". Payment systems involving state-issued currencies have been used for centuries to facilitate transactions, and the involvement of the state has been critical as state institutions can dampen fluctuations in the value of the currency. [?] Controlling money supply is critical to ensure stable prices that facilitate trade [?] instead of speculation.[?]

As transactions on the Internet, such as sending an e-mail or reading a Web site, tend to be of smaller value than traditional transactions involving the exchange of physical goods, we are faced with the challenge of reducing the mental and technical overheads of existing payment systems to handle micropayments. Addressing this problem is urgent: ad-blocking technology is eroding advertising as a substitute for micropayments [?], and the Big Data business model where citizens pay with their private information [?] in combination with the deep state hastens our society's regression towards post-democracy [?].

The focus of this paper is on Taler, a new free software payment system designed to meet certain key ethical considerations. In Taler, the customer remains anonymous, while the merchant is taxable. Here, *anonymous* simply means that the payment system does not require any personal information from the customer, and that different transactions by the same customer are unlinkable. Naturally, the specifics of the transaction—such as delivery of goods to a shipping address, or the use of non-anonymous IP-based communication—may still leak information about the customer's identity. *Taxable* means that the state can obtain the necessary information about the contract to levy income, sales or value-added taxes. Taler uses blind signatures [?] to create digital coins, and a new "refresh" protocol to allow giving change and refunds while maintaining unlinkability.

This paper will not consider the details of Taler's cryptographic protocols[1], as for usability one needs to completely hide the cryptography from the users. Thus, this paper will focus on an analytical description of how to achieve usable and secure electronic payments. Our focus is to show that existing *mental models* users have from existing widespread payment systems will apply naturally. We leave a usability study with actual users for future work, as we believe that the basic architecture of such a system is sufficiently interesting by itself.

Key contributions of this paper are:

- A description of different payment systems using common terminology, allowing us to analytically compare these systems with respect to security and usability.
- An introduction to the Taler payment system from the perspective of users and merchants, with a focus on how to achieve secure payments in a way that is intuitive and has adequate fail-safes.
- Detailed considerations for how to adapt Taler to Web payments and the intricacies of securing payments within the constraints of modern "secure" browsers.
- A publicly available free software reference implementation of the proposed architecture.

[1]No citation given due to anonymous submission.

2. EXISTING PAYMENT WORKFLOWS

Before we look at the payment workflow for Taler, we will sketch the workflow of existing payment systems. This will establish a common terminology, a baseline for comparison and crucially also an indication as to how we can relate Taler's workflow to existing *mental models* that users already have, thereby allowing us to judge the mental adaptation costs required to transition to transactions with Taler.

2.1 Cash

Cash has traditionally circulated by being passed directly from buyers to sellers, with each seller then becoming a buyer. Thus, cash is inherently a *peer-to-peer* payment system, as participants all appear in the both buyer and seller roles, merely at different times. However, this view is both simplified and somewhat dated.

In today's practice, cash is frequently first *withdrawn* from ATMs by customers, who then *spend* it with merchants, who finally *deposit* the cash with their respective *bank*. In this flow, security is achieved as the customer *authenticates* to the ATM using *credentials* provided by the customer's bank, and the merchant specifies his *account* details when depositing the cash. The customer does not authenticate when spending the cash, but the merchant *validates* the authenticity of the *coins* or bills. Coins and bills are *minted* by state-licensed institutions, such as the US Mint. These institutions also provide detailed instructions for how to validate the authenticity of the coins or bills [?], and are typically the final trusted authority on the authenticity of coins and bills.

As customers need not authenticate, purchases remain *anonymous*, modulo the limited tracking enabled by serial numbers printed on bills. [?]

Spending cash does not provide any inherent *proof of purchase* for the customer, instead the merchant may provide paper *receipts* which are generated independently and do not receive the same anti-forgery protections that are in place for cash.

Against most attacks customers and merchants limit their risks to the amount of cash they carry or accept at a given time [?]. Additionally, customers are advised to choose the ATMs they use carefully, as malicious ATMs may attempt to steal their customer's credentials. Authentication with an ATM can involve a special ATM card, or more commonly the use of credit or debit cards. In all these cases, these physical security tokens are issued by the customer's bank of the customer.

2.2 Credit and debit cards

Credit and debit card payments operate by the customer providing their credentials to the merchant. Many different authentication and authorization schemes are in use in various combinations, including both secret information, usually PINs, and physical security devices like TANs [?] (cards with an EMV chip [?]), and the customer's mobile phone [?]. A typical modern Web payment process involves **(0)** the merchant offering a "secure" communication channel using TLS based on the X.509 public key infrastructure,[2] **(1)** selecting a *payment method*, **(2)** entering the credit card details like owner's name, card number, expiration time, CVV code, and billing address, **(3)** (optionally) authorizing the transaction via mobile TAN, or by authenticating against the customer's bank, often on a Web site that is operated by the payment processor and *not* the customer's bank. Figure 1 shows a typical card-based payment process on the Web today using the UML style of the W3c payment interest group [?]. Most of the details are not relevant to this paper, but the figure nicely illustrates the complexity of the common 3-D secure (3DS) process.

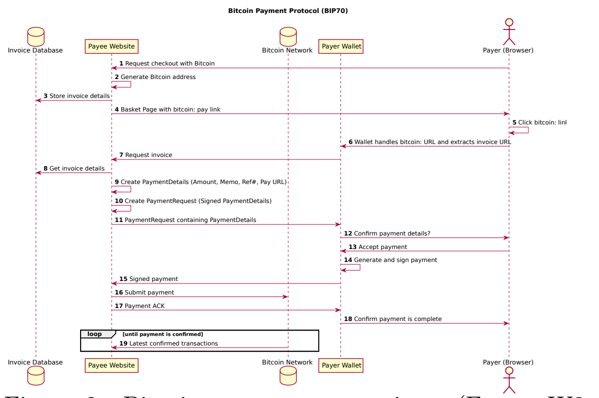
Given this process, there is an inherent risk of information leakage of customers' credentials. Fraud detection systems attempt to detect misuse of leaked credentials, and payment system providers handle disputes between customers and merchants. As a result, Web payment processes may finish with **(4)** the payment being rejected for a variety of reasons, from false-positives in fraud detection to the merchant not accepting the particular card issuer.

Traditionally, merchants bear most of the financial risk, and a key "feature" of the 3DS process compared to traditional card payments is hence to shift dispute liability to the issuer of the card, who may then shift it to the customer. Even in cases where the issuer or the merchant remain legally first in line, there are still risks customers incur from the card dispute procedures, such as neither they not the payment processor noticing fraudulent transactions, or them noticing fraudulent transactions past the date at which their bank will refund them. The customer also typically only has a merchant-generated comment and the amount paid in his credit card statement as a proof for the transaction. Thus, the use of credit cards online does not generate any verifiable electronic receipts for the customers, enabling malicious merchants to later change the terms of the contract. Beyond these issues, customers face secondary risks of identity theft from the personal details exposed by the authentication procedures. In this case, even if the financial damages are ultimately covered by the bank, the customer always has to deal with the hassle of notifying the bank in the first place. As a result, customers must remain wary about their card use, which limits their online shopping [?, p. 50].

2.3 Bitcoin

Bitcoin operates by recording all transactions in a pseudonymous public *ledger*. A Bitcoin account is identified by its public key and the owner(s) must know the corresponding private key, which in turn is used to authorize the transfer of Bitcoins from the account to other accounts. The information in the global public ledger allows everybody to compute the balances in all accounts and to see all transactions. Transactions are denom-

[2]Given numerous TLS protocol and implementation flaws as well as X.509 key managment incidents in recent years [?], the security provided by TLS is at best questionable.


Bitcoin Payment Protocol (BIP70)
Invoice Database
Payee Website
Bitcoin Network
Payer Wallet
Payer (Browser)
1 Request checkout with Bitcoin
2 Generate Bitcoin address
3 Store invoice details
4 Basket Page with bitcoin: pay link
5 Click bitcoin: linl
6 Wallet handles bitcoin: URL and extracts invoice URL
7 Request invoice
8 Get invoice details
9 Create PaymentDetails (Amount, Memo, Ref#, Pay URL)
10 Create PaymentRequest (Signed PaymentDetails)
11 PaymentRequest containing PaymentDetails
12 Confirm payment details?
13 Accept payment
14 Generate and sign payment
15 Signed payment
16 Submit payment
17 Payment ACK
18 Confirm payment is complete
loop
[until payment is confirmed]
19 Latest confirmed transactions
Invoice Database
Payee Website
Bitcoin Network
Payer Wallet
Payer (Browser)
Figure 2: Bitcoin payment processing. (From: W3c Web Payments IG.)

inated in a new currency labeled BTC, whose valuation depends upon speculation. Adding transactions to the global public ledger involves broadcasting the transaction data, peers verifying and appending it to the public ledger, and some peer in the network solving a moderately hard computational proof-of-work puzzle; the latter process is called *mining*. The mining process is incentivised by transaction fees and mining rewards, the latter also providing the process of initial accumulation for BTC. [?] Conversion to and from BTC from and to other currencies incurs substantial fees [?]. There is now an extreme diversity of Bitcoin-related payment technologies, but usability improvements are usually achieved by adding a "trusted" third party, and there have been many incidents where such parties then embezzled funds from their customers. The classical Bitcoin payment workflow consisted of entering payment details into a peer-to-peer application. The user would access his Bitcoin *wallet* and instruct it to transfer a particular amount from one of his accounts to the account of the merchant, possibly including additional metadata to be associated with the transfer and embedded into the global public ledger. The wallet application would then transmit the request to the Bitcoin peer-to-peer overlay network. The use of an external payment application makes wallet-based payments significantly less browser-friendly than ordinary card payments, as illustrated in Figure 2.

Bitcoin payments are only confirmed when they appear in the public ledger, which is updated at an average frequency of once every 10 minutes. Even then, it is possible that a fork in the so-called block chain may void durability of the transaction [?]. As a result, customers and merchants must either accommodate this delay, or incur fraud risks during this indetermined period.

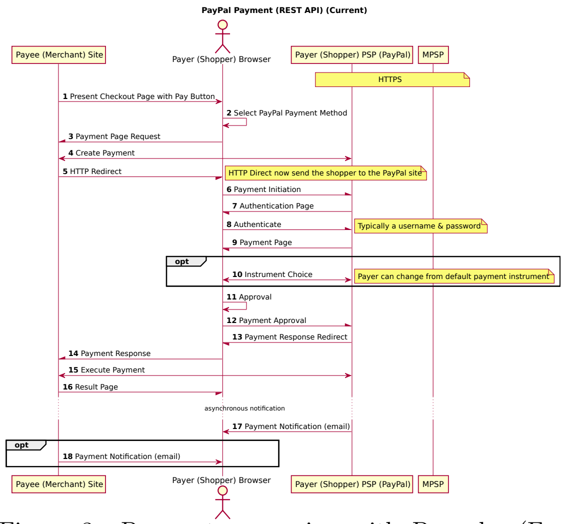
Bitcoin is considered to be secure against an adversary who cannot control around a fifth of the Bitcoin miner's computational resources [?, ?, ?]. As a result, the network must expend considerable computational resources to keep this value high. In fact, "a single Bitcoin transaction uses roughly enough electricity to power 1.57 American households for a day". [?] At present, these costs are largely hidden by speculation in BTC, but that speculation itself contributes to BTC being unstable. [?, ?, ?].

There are several examples of Bitcoin's pseudononymity being broken by investigators [?]. Mixnets afford protection against this, but they require numerous transactions, exacerbating Bitcoin's already high transaction costs. Bitcoin's pseudononymity applies equally to both customers and merchants, making Bitcoin highly amenable to tax evasion, money laundering, and sales of contraband. As a result, anonymity tools like mixnets do not enjoy particularly widespread support in the Bitcoin community where many participants seek to make the currency appear more legitimate.

2.4 Walled garden payment systems

Walled garden payment systems offer ease of use by processing payments using a trusted payment service provider. Here, the customer authenticates to the trusted service and instructs the payment provider to execute a transaction on his behalf (Figure 3). In these payment systems, the provider basically acts like a bank with accounts carrying balances for the various users. In contrast to traditional banking systems, both customer and merchant are forced to have an account with the same provider. Each user must take the effort to establish his identity with a service provider to create an account. Merchants and customers obtain the best interoperability in return for their account creation efforts if they start with the biggest providers. As a result, there are a few dominating walled garden providers, with AliPay, ApplePay, GooglePay, SamsungPay and PayPal being the current oligopoly. In this paper, we will use PayPal as a representative example for our discussion of these payment systems.

As with card payments, these oligopolies are politically dangerous [?] and the lack of competition can result in excessive profit taking that may require political solutions [?] to the resulting market failure. The use of


PayPal Payment (REST API) (Current)
Payee (Merchant) Site
Payer (Shopper) Browser
Payer (Shopper) PSP (PayPal)
MPSP
HTTPS
1 Present Checkout Page with Pay Button
2 Select PayPal Payment Method
3 Payment Page Request
4 Create Payment
5 HTTP Redirect
HTTP Direct now send the shopper to the PayPal site
6 Payment Initiation
7 Authentication Page
8 Authenticate
Typically a username & password
9 Payment Page
opt
10 Instrument Choice
Payer can change from default payment instrument
11 Approval
12 Payment Approval
13 Payment Response Redirect
14 Payment Response
15 Execute Payment
16 Result Page
asynchronous notification
17 Payment Notification (email)
opt
18 Payment Notification (email)
Payee (Merchant) Site
Payer (Shopper) Browser
Payer (Shopper) PSP (PayPal)
MPSP
Figure 3: Payment processing with Paypal. (From: W3c Web Payments IG.)

non-standard proprietary interfaces to the payment processing service of these providers serves to reinforce the customer lock-in.

3. TALER

Taler is a free software cryptographic payment system with an open protocol specification that couples cash-like anonymity for customers when they spend money with low transaction costs, signed digital receipts, and accurate income information to facilitate taxation and anti-corruption efforts.

Taler achieves anonymity for buyers using *blind signatures* [?]. Ever since their discovery thirty years ago, cryptographers have viewed blind signatures as the optimal cryptographic primitive for consumer level transaction systems. Our goal is for Taler to become the first transaction system based on blind signatures to see widespread adoption. Hiding the cryptography from users and integrating smoothly with the Web are central components of our technical strategy to achieve this.

There are four components of the Taler system (Figure 4):

- *Wallets* are software packages that allow customers to withdraw, hold, and spend coins. Wallets also manage the customer's reserve accounts, and keep receipts in a transaction history. Wallets can be realized as browser extensions, mobile Apps or even in custom hardware.

- *Exchanges* enable customers to withdraw anonymous digital coins and merchants to deposit digital coins, in exchange for bank money. Exchanges learn the amounts withdrawn by customers and deposited by merchants, but they do not learn the relationship between customers and merchants. Exchanges perform online detection of double spending, thus providing merchants instant feedback, —including digital proofs—in case of misbehaving customers.

- *Merchants* provide goods or services in exchange for coins held by customers' wallets. Merchants deposit these coins at the exchange for their regular currency value. Merchants consist of a *frontend* which interacts with the customer's wallet, and a *backend* that interacts with the exchange. Typical frontends include Web shops and point-of-sale systems.

- *Auditors* verify that exchanges operate correctly to limit the risk that customers and merchants incur by using a particular exchange.

The specific protocol between wallet and merchant depends on the setting. For a traditional store, an NFC protocol might be used between a point-of-sale system and a mobile application. In this paper, we focus on Web payments for an online shop.

3.1 Web payment workflow

We explain how the actors in the Taler system interact by documenting a typical payment.

Initially, the customer must once install the Taler wallet extension for his browser. Naturally, this step may become superfluous if Taler is integrated tightly with browsers in the future. Regardless, installing the extension involves one or two clicks to confirm the operation once the user was pointed to the correct Web site. Restarting the browser is not required.

Withdrawing coins

As with cash, the customer must first withdraw digital coins (Figure 5). For this, the customer must first visit the online banking portal of his bank. Here, the bank will typically require some form of authentication, the specific method used depends on the bank (Figure 6a).

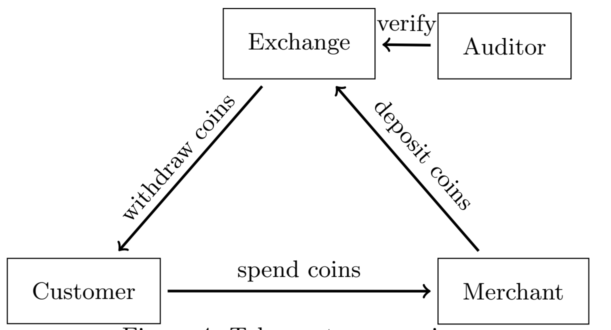
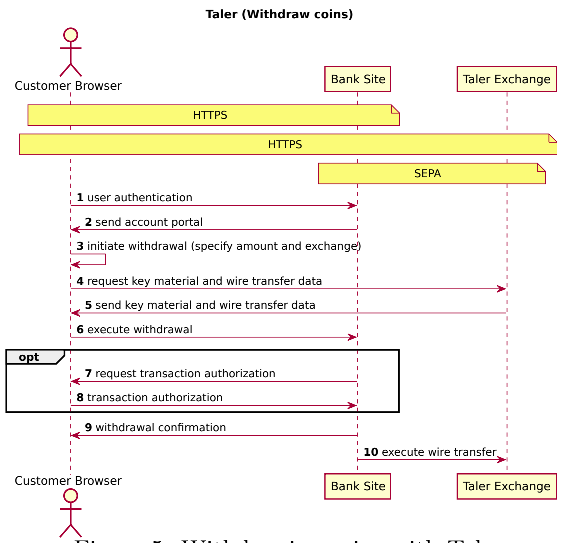
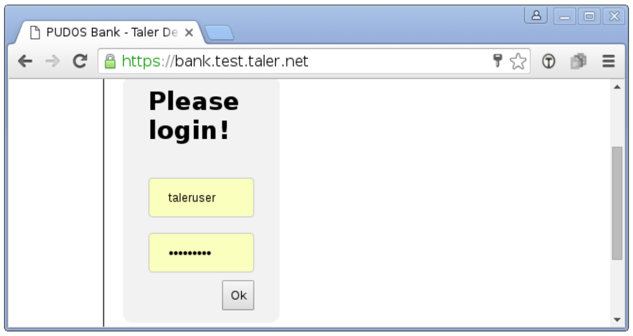


Figure 4: Taler system overview.



Taler (Withdraw coins)
Customer Browser
Bank Site
Taler Exchange
HTTPS
HTTPS
SEPA
1 user authentication
2 send account portal
3 initiate withdrawal (specify amount and exchange)
4 request key material and wire transfer data
5 send key material and wire transfer data
6 execute withdrawal
opt
7 request transaction authorization
8 transaction authorization
9 withdrawal confirmation
10 execute wire transfer
Customer Browser
Bank Site
Taler Exchange

Figure 5: Withdrawing coins with Taler.



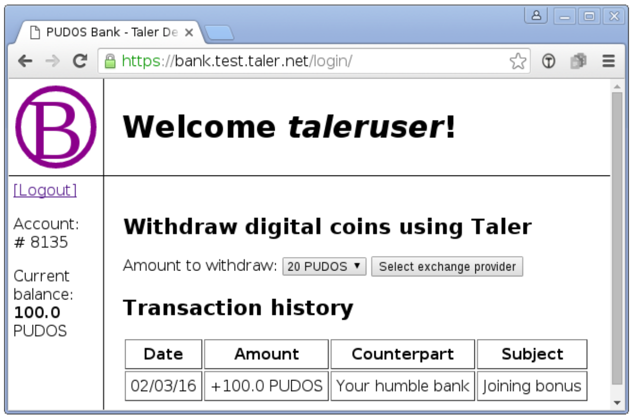
Please login!

taleruser

•••••••••

Ok

(a) Bank login. (Simplified for demonstration.)



Welcome *taleruser*!

[Logout]

Account: # 8135

Current balance: **100.0** PUDOS

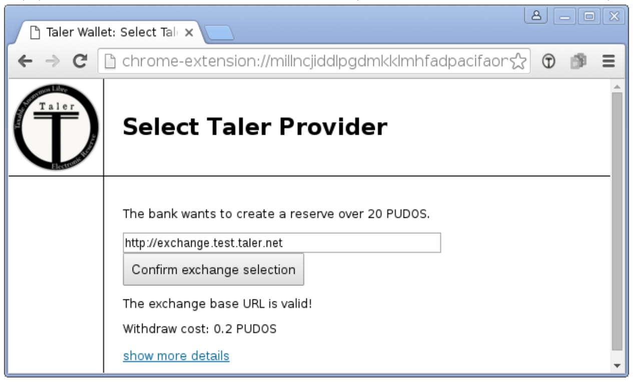
Withdraw digital coins using Taler

Amount to withdraw: 20 PUDOS ▾ Select exchange provider

Transaction history

Date	Amount	Counterpart	Subject
02/03/16	+100.0 PUDOS	Your humble bank	Joining bonus

(b) Specify amount to withdraw. (Integrated bank support.)



Select Taler Provider

The bank wants to create a reserve over 20 PUDOS.

http://exchange.test.taler.net

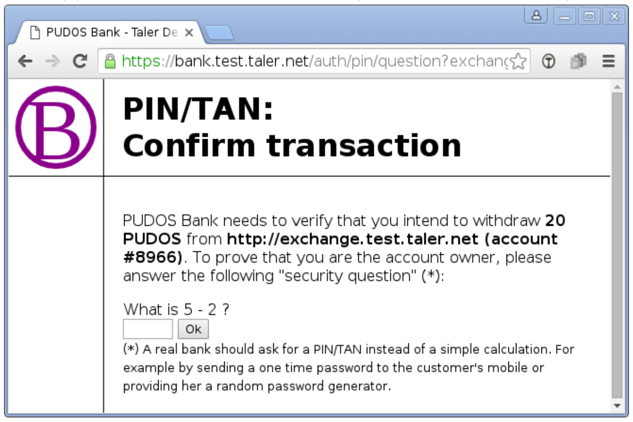
Confirm exchange selection

The exchange base URL is valid!

Withdraw cost: 0.2 PUDOS

show more details

(c) Select exchange provider. (Generated by wallet.)



PIN/TAN: Confirm transaction

PUDOS Bank needs to verify that you intend to withdraw **20 PUDOS** from **http://exchange.test.taler.net (account #8966)**. To prove that you are the account owner, please answer the following "security question" (*):

What is 5 - 2 ?

Ok

(*) A real bank should ask for a PIN/TAN instead of a simple calculation. For example by sending a one time password to the customer's mobile or providing her a random password generator.

(d) Confirm transaction with a PIN. (Generated by bank.)

Figure 6: Required steps in a Taler withdrawal process.

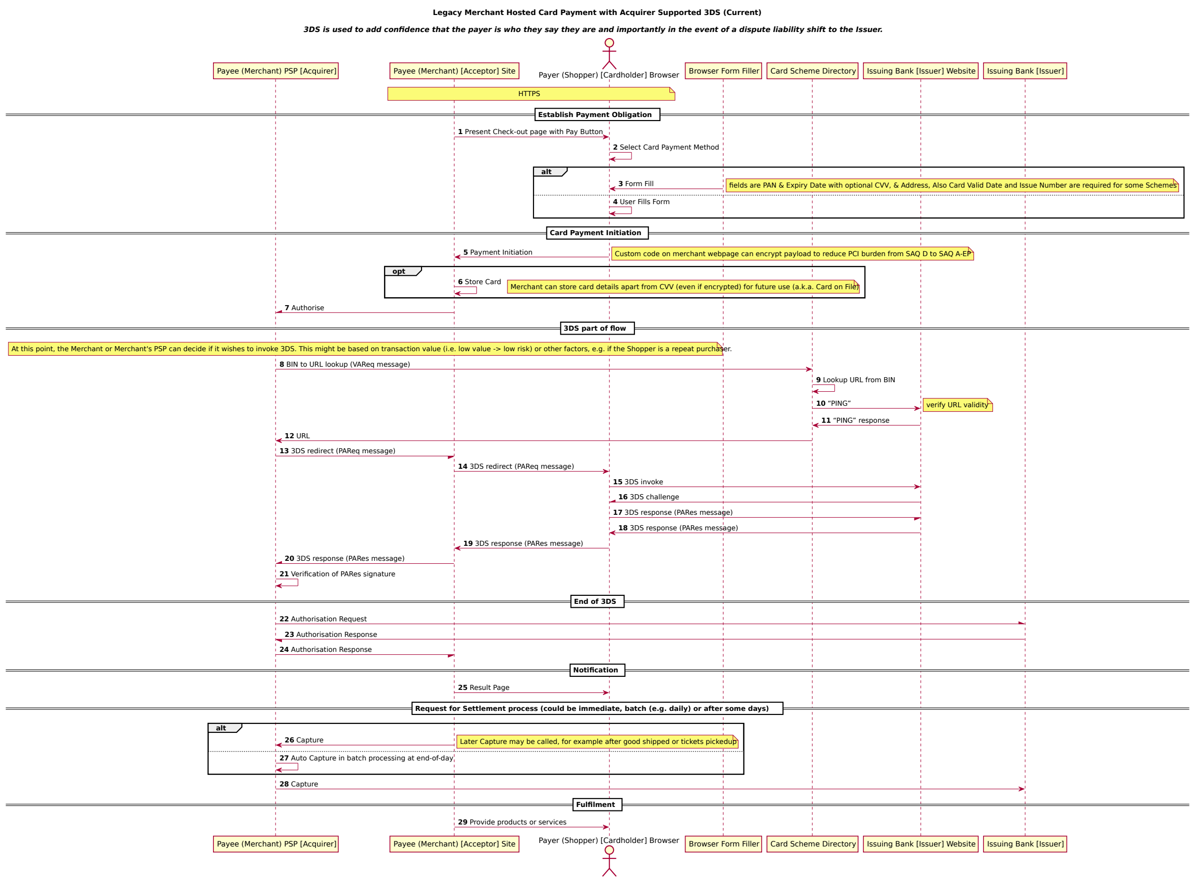

Legacy Merchant Hosted Card Payment with Acquirer Supported 3DS (Current)
3DS is used to add confidence that the payer is who they say they are and importantly in the event of a dispute liability shift to the Issuer.
Payee (Merchant) PSP [Acquirer]
Payee (Merchant) [Acceptor] Site
Payer (Shopper) [Cardholder] Browser
Browser Form Filler
Card Scheme Directory
Issuing Bank [Issuer] Website
Issuing Bank [Issuer]
HTTPS
Establish Payment Obligation
1 Present Check-out page with Pay Button
2 Select Card Payment Method
alt
3 Form Fill
fields are PAN & Expiry Date with optional CVV, & Address, Also Card Valid Date and Issue Number are required for some Schemes
4 User Fills Form
Card Payment Initiation
5 Payment Initiation
Custom code on merchant webpage can encrypt payload to reduce PCI burden from SAQ D to SAQ A-EP
opt
6 Store Card
Merchant can store card details apart from CVV (even if encrypted) for future use (a.k.a. Card on File)
7 Authorise
3DS part of flow
At this point, the Merchant or Merchant's PSP can decide if it wishes to invoke 3DS. This might be based on transaction value (i.e. low value -> low risk) or other factors, e.g. if the Shopper is a repeat purchaser.
8 BIN to URL lookup (VAReq message)
9 Lookup URL from BIN
10 "PING"
verify URL validity
11 "PING" response
12 URL
13 3DS redirect (PAReq message)
14 3DS redirect (PAReq message)
15 3DS invoke
16 3DS challenge
17 3DS response (PARes message)
18 3DS response (PARes message)
19 3DS response (PARes message)
20 3DS response (PARes message)
21 Verification of PARes signature
End of 3DS
22 Authorisation Request
23 Authorisation Response
24 Authorisation Response
Notification
25 Result Page
Request for Settlement process (could be immediate, batch (e.g. daily) or after some days)
alt
26 Capture
Later Capture may be called, for example after good shipped or tickets pickedup
27 Auto Capture in batch processing at end-of-day
28 Capture
Fulfilment
29 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]

Figure 1: Card payment processing with 3DS. (From: W3c Web Payments IG.)

The next step depends on the Taler support offered by the bank:

- If the bank does not properly integrate with Taler, the customer needs use the menu of the wallet to create a *reserve*. The wallet will ask which amount in which *currency* (i.e. EUR or USD) the customer wants to withdraw and allow the customer to select an exchange. Given this information, the wallet will instruct the customer to transfer the respective amount to the account of the exchange. The customer will have to enter a 54-character reserve key which includes 256 bits of entropy and an 8-bit checksum into the transfer subject. Naturally, this is exactly the kind of interaction we would like to avoid for usability.
- Hence, if the bank properly integrates with Taler, the customer has a form in the online banking portal where he can specify an amount to withdraw (Figure 6b). The bank then triggers an interaction with the wallet to allow the customer to select an exchange (Figure 6c). Afterwards, the wallet instructs the bank about the details of the wire transfer. The bank asks the customer to authorize the transfer, and finally confirms to the wallet that the transfer has been successfully initiated.

In either case, the wallet can then withdraw the coins from the exchange, and does so in the background without further interaction with the customer.

In principle, the exchange can be directly operated by the bank, in which case the step where the customer selects an exchange may be skipped by default. However, we generally assume that the exchange is a separate entity, as this yields the largest anonymity set for customers and may help create a competitive market.

Spending coins

At a later point in time, the customer can spend his coins by visiting a merchant that accepts digital coins in the respective currrency issued by the respective exchange (Figure 7). Merchants are generally configured to either accept a specific exchange, or to accept all the exchanges audited by a particular auditor. Merchants can also set a ceiling for the maximum amount of transaction fees they are willing to cover. Usually these details should not matter for the customer, as we expect most merchants to allow most accredited exchange providers, and for exchanges to operate with transaction fees acceptable to most merchants. If transaction fees are higher than what is covered by the merchant, the customer may choose to cover them.

As with traditional Web transactions, the customer first selects which items he wishes to buy. This can involve building a traditional shopping cart, or simply clicking on a particular link for the respective article (Figure 8a). As with card payments, the Web shop may then allow the customer to select a payment method, including Taler. However, Taler also allows the Web shop to detect the presence of a Taler wallet (Figure 9), so that this step may be skipped (as it is in Figure 8). If Taler was detected or selected, the Web shop sends a digitally signed *contract proposal* to the wallet extension (Figure 10). The wallet then presents the contract to the user. The format of the contract is in an extensible JSON-based format defined by Taler and not HTML, as the rendering of the contract is done by the wallet to ensure correct visual representation. In the case that transaction fees need to be covered by the customer, these are shown together with the rest of the proposed contract.

If the customer approves the contract by clicking the "Confirm Payment" button (Figure 8b), his wallet signs the contract with enough coins to cover the contract's cost, stores all of the information in its local database, and redirects the browser to a *fulfillment* URL provided by the merchant (Figure 8c). The wallet cannot directly send the payment to the merchant, as the page showing the contract is provided as a background page controlled by the Web Extension[3] and thus submitting coins from the background would not use the HTTP-context (such as cookies) that the Web shop's page requires for session management.

Instead, the server-side of the fulfillment page usually first detects that the contract has not yet been paid by checking the merchant's local database and the HTTP session state. **(A)** If the state indicates that this customer did not yet pay, the merchant generates a page that shows the customer an indication that the payment is being processed, and tries to interact with the wallet, requesting payment. If the wallet is not detected after a few milliseconds, the page transitions to the card payment process. If the wallet is present, the page requests

[3] `https://developer.chrome.com/extensions`

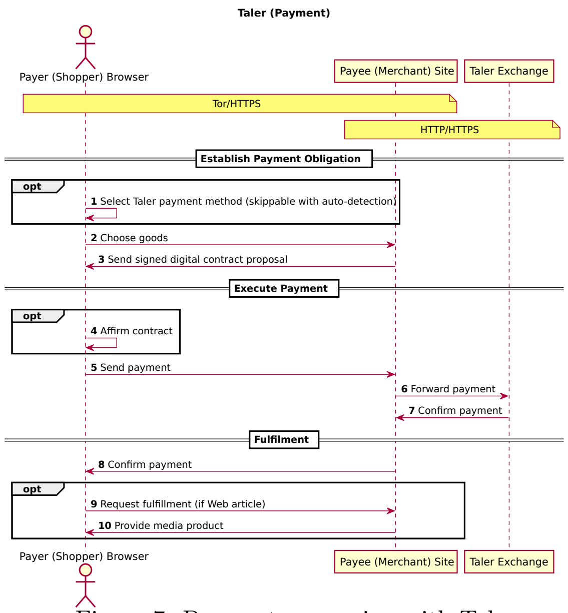
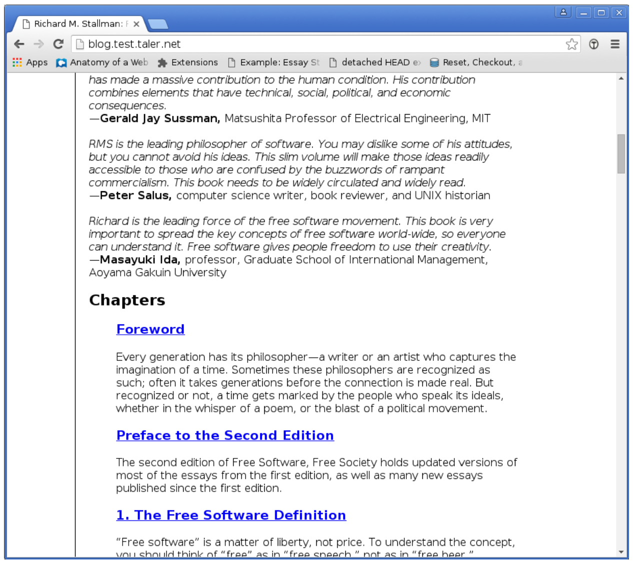


Figure 7: Payment processing with Taler.



has made a massive contribution to the human condition. His contribution combines elements that have technical, social, political, and economic consequences.
—**Gerald Jay Sussman,** Matsushita Professor of Electrical Engineering, MIT

RMS is the leading philosopher of software. You may dislike some of his attitudes, but you cannot avoid his ideas. This slim volume will make those ideas readily accessible to those who are confused by the buzzwords of rampant commercialism. This book needs to be widely circulated and widely read.
—**Peter Salus,** computer science writer, book reviewer, and UNIX historian

Richard is the leading force of the free software movement. This book is very important to spread the key concepts of free software world-wide, so everyone can understand it. Free software gives people freedom to use their creativity.
—**Masayuki Ida,** professor, Graduate School of International Management, Aoyama Gakuin University

Chapters

Foreword

Every generation has its philosopher—a writer or an artist who captures the imagination of a time. Sometimes these philosophers are recognized as such; often it takes generations before the connection is made real. But recognized or not, a time gets marked by the people who speak its ideals, whether in the whisper of a poem, or the blast of a political movement.

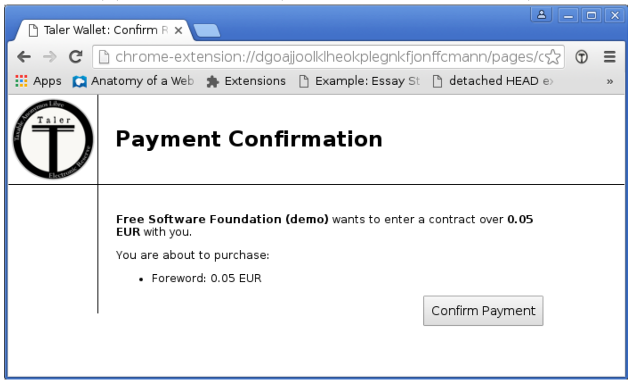
Preface to the Second Edition

The second edition of Free Software, Free Society holds updated versions of most of the essays from the first edition, as well as many new essays published since the first edition.

1. The Free Software Definition

"Free software" is a matter of liberty, not price. To understand the concept, you should think of "free" as in "free speech," not as in "free beer."

(a) Select article. (Generated by Web shop.)



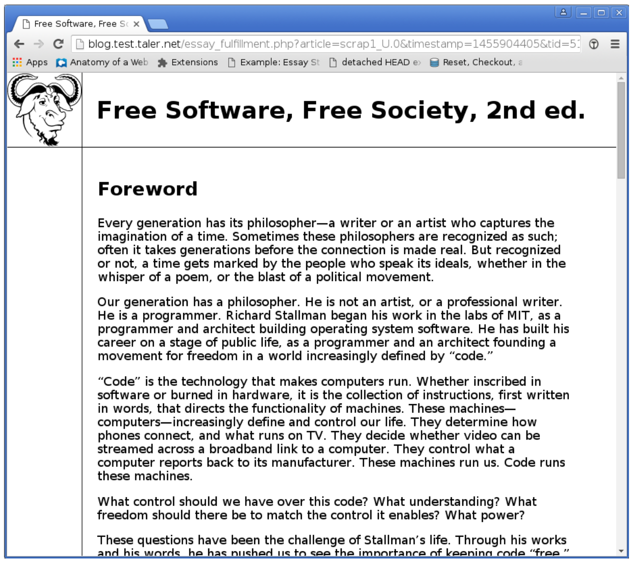
Payment Confirmation

Free Software Foundation (demo) wants to enter a contract over **0.05 EUR** with you.

You are about to purchase:

- Foreword: 0.05 EUR

(b) Confirm payment. (Generated by Taler wallet.)



Free Software, Free Society, 2nd ed.

Foreword

Every generation has its philosopher—a writer or an artist who captures the imagination of a time. Sometimes these philosophers are recognized as such; often it takes generations before the connection is made real. But recognized or not, a time gets marked by the people who speak its ideals, whether in the whisper of a poem, or the blast of a political movement.

Our generation has a philosopher. He is not an artist, or a professional writer. He is a programmer. Richard Stallman began his work in the labs of MIT, as a programmer and architect building operating system software. He has built his career on a stage of public life, as a programmer and an architect founding a movement for freedom in a world increasingly defined by "code."

"Code" is the technology that makes computers run. Whether inscribed in software or burned in hardware, it is the collection of instructions, first written in words, that directs the functionality of machines. These machines—computers—increasingly define and control our life. They determine how phones connect, and what runs on TV. They decide whether video can be streamed across a broadband link to a computer. They control what a computer reports back to its manufacturer. These machines run us. Code runs these machines.

What control should we have over this code? What understanding? What freedom should there be to match the control it enables? What power?

These questions have been the challenge of Stallman's life. Through his works and his words, he has pushed us to see the importance of keeping code "free."

(c) Receive article. (Generated by Web shop.)
Figure 8: Required steps in a Taler checkout process.

payment from the wallet. The wallet then determines that the customer already confirmed the payment and immediately transfers the coins to the JavaScript logic of the fulfillment page. The fulfillment page then transfers the coins to the merchant, usually using an asynchronous HTTP POST request. The request is controlled by the merchant's JavaScript and not by the wallet. This ensures that the merchant is in full control of the communication between the merchant's server and the client-side scripts interacting with the merchant's server. The interactions with the wallet are thus purely local interactions within the browser. Upon receipt of the payment information, the merchant confirms the payment with the exchange, marks the payment as received, and notifies the JavaScript on the server side of the result.

- If the payment fails on the network, the request is typically retried. How often the client retries automatically before informing the user of the network issue is up to the merchant. If the network failure persists and is between customer and merchant, the wallet will try to recover control over the coins at the exchange by effectively spending the coins first using Taler's special "refresh" protocol. In this case, later deposits by the merchant will simply fail. If the merchant already succeeded with the payment before the network failure, the customer can either retry the operation later via the transaction history, or demand a refund (see below). Handling these errors does not require the customer to give up his privacy.

- If the payment fails due to the exchange claiming that the request was invalid, the diagnostics created by the exchange are passed to the wallet for inspection. The wallet then decides whether the exchange was correct, and can then inform the user about a fraudulent or buggy exchange. At this time, it allows the user to export the relevant cryptographic data to be used in court. If the exchange's proofs were correct and coins were double-spent, the wallet informs the user that its database must have been out-of-date, updates the database and allows the user to retry the transaction.

- If the payment succeeded, the JavaScript on the client side triggers effectively a "reload" of the fulfillment page, triggering case (B) detailed below.

(B) Upon subsequent visits, the server detects that the payment has already been processed and directly generates a fulfillment page either confirming the payment, or—in the case of payments for a digital article—transmits the digital artifact to the client.

Bookmarks and deep links

This particular architecture also enables smooth use of the payment URIs on the contemporary Web. In particular, we need to consider the possibility that a user may bookmark the fulfillment page, or forward a link to the fulfillment page to another user.

The given design supports *bookmarking*. If the merchant's session management is still tracking the user when he returns via the bookmark, the page generation detects that the user has already paid and serves the final fulfillment page. If the session has been lost, the merchant will generate a fulfillment page asking for payment. In this case, the wallet will detect that it has already paid this contract via a unique identifier in the contract, and will automatically re-play the payment. The merchant confirms that this customer already paid, and generates the final fullfilment page that the user has previously payed for (and seen). All this still appears as instantaneous to the user as it merely adds a few extra network round trips.

In contrast, if the customer sends a link to the fulfillment page to another user, thereby possibly sharing a *deep link* into the merchant's shop, the other customer's wallet will fail to find an existing payment. In this case, the fulfillment page will thus not immediately receive the payment details and instead provide the user with the proposed contract which contains a description of the item(s) previoulsy bought by the other user. Then the recipient of the link can decide to purchase the same item(s).

The design, in particular POSTing the coins asynchronously from JavaScript, also ensures that the user can freely navigate with the back and forward buttons. As all requests from all HTTP(S) URIs ever seen by the user in the browser are fetched via HTTP GET, they can be bookmarked, shared and safely reloaded. For caching, the merchant needs to ensure that the fulfillment page generated in case (A) is not cached by the browser, and in case (B) is not cached in the network.

As an aside, there are actually several distinct roles comprising the merchant: shopping pages end their role by proposing a contract, while a fulfillment page begins its life processing a contract. It is thus possible for these components being managed by seperate parties. The control of the fulfillment page over the transmission of the payment information minimizes the need for exceptions to handle cross-origin resource sharing. [**?**, **?**]

Giving change and refunds

An important technical difference between Taler and previous transaction systems based on blind signing is that Taler is able to provide unlinkable change and refunds. From the user's point of view, obtaining change is automatic and handled by the wallet, i.e. if the user has a single coin worth €5 and wants to spend €2, the wallet may request three €1 coins in change — critically, this is completely hidden from the user. In fact, the graphical user interface does not offer a way to inspect the denominations of the various coins in the wallet, it only shows the total amount availabe in each denomination. Expanding the views to show details may show the exchange providers and fee structure, but not the cryptographic coins. Consequently, the major cryptographic advances of Taler are invisible to the user.

Taler's technology also allows merchants to give refunds to customers. For this, the merchant merely has to send a signed message to the exchange confirming the refund, and notify the customer's wallet that the respective transaction was refunded. This can even be done with anonymous customers, as refunds are given as additional change to the owner of the coins that were originally spent to pay for the refunded transaction.

Taler's protocol ensures unlinkability for both changes and refunds, thus assuring that the user has key conveniences of other payment systems, while maintaining the security standard of an anonymous payment system.

3.2 NFC payments

We have so far focused on how Taler would be used for Web payments; however, Taler can also be naturally used over other protocols, such as near field communication (NFC). Here, the user would hold his NFC-enabled device running a wallet application near an NFC terminal to obtain the contract, confirm the payment on his device, which would then transfer the coins and obtain a receipt. Given that an NFC application would be less restricted in its interaction with the point-of-sale system compared to the complex security model of modern browsers, running Taler over NFC is largely a simplfication.

There are no significant new concerns arising from an NFC device possibly losing contact with a point-of-sale system. Already for Web payments, Taler employs only idempotent operations to ensure coins are never lost and that transactions adequately persist even in the case of network or endpoint failures. As a result, the NFC system can simply use the same transaction models to replay transmissions once contact with the point-of-sale system is reestablished.

3.3 Peer-to-peer payments

Peer-to-peer payments are possible with Taler as well; however, we need to distinguish two types of peer-to-peer payments.

First, there is the *sharing* of coins among entities that mutually trust each other, for example within a family. Here, all the users have to do is to export and import electronic coins over a secure channel, such as encrypted e-mail or via NFC. For NFC, the situation is pretty trivial, while secure communication over the Internet is likely to remain a significant usability challenge. We note that sharing coins by copying the respective private keys across devices is not taxable: the exchange is not involved, no contracts are signed, and no records for taxation are created. However, the involved entities must trust each other, as after copying a private key both parties could spend the coins. Given this crucial limitation inherent in sharing keys, we consider it ethically acceptable that sharing is not taxable.

Second, there is the *transactional* mutually exclusive transfer of ownership. This requires the receiving party to have a *reserve* with an exchange, and the exchanges would have to support wire transfers among them. If taxability is desired, the *reserve* would still need to be tied to a particular citizen's identity for tax purposes, and thus require similar identification protocols as commonly used for establishing a bank account. Thus, in terms of institutions, one would expect this setup to be offered most easily by traditional banks. In terms of usability, transactional transfers are just as easy as sharing when performed over NFC, but more user friendly when performed over the Internet as they do not require a secure communication channel: the Taler protocol is by design still safe to use even if the communication is made over an unencrypted channel. Only the authenticity of the proposed contract needs to be assured.

3.4 Usability for merchants

Payment system security and usability is not primarily a concern for customers, but also for merchants. For consumers, existing schemes may be inconvenient and not provide privacy, but remembering to protect a physical token (i.e. the card) and to guard a secret (i.e. the PIN) is relatively straightforward. In contrast, merchants are expected to "securely" handle sensitive customer payment data on networked computing devices. However, securing computer systems—and especially payment systems that represent substantial value—is a hard challenge, as evidenced by large-scale break-ins with millions of consumer card records being illicitly copied. [**?**]

Thus, we cannot ignore the usability at the merchant site when trying to understand the usability of a payment system, especially as for deployment we will have to convince millions of merchants that the Taler system is advantageous. The high-level cryptographic design already provides the first major advantage, as merchants do never receive sensitive payment-related customer information. Thus, they do not have to be subjected to costly audits or certified hardware, as is commonly the case for processing card payments. [**?**] In fact, the exchange does not need to have a formal business relationship with the shop at all. According to our design, the exchange's contract with the state regulator or auditor and the customers ought to state that it must honor all (legal and valid) deposits it receives. Hence, a merchant supplying a valid deposit request should be able to enforce this in court without a prior business agreement with the exchange. This dramatically simplifies setting up a shop, to the point that the respective software only needs to be provided with the merchant's wire transfer routing information to become operational.

Figure 9 shows how easy it is for a Web shop to detect the presence of a Taler wallet. This leaves a few cryptographic operations, such as signing a contract and verifying the customer's and the exchange's signatures, storing transaction data as well as matching sales with incoming wire transfers from the exchange. Taler simplifies this for merchants by providing a generic payment processing *backend* for the Web shops.

Figure 11 shows how the secure payment components interact with the existing Web shop logic. First, the Web shop frontend is responsible for constructing the shopping cart. For this, the shop frontend generates the usual Web pages which are shown to the user's browser client frontend. Once the order has been constructed, the shop frontend gives a *proposed contract* in JSON format to the payment backend, which signs it and returns it to the frontend. The frontend then transfers the signed contract over the network, and passes it to the wallet (sample code for this is in Figure 10). Here, the wallet operates from a secure *background* on the client side, which allows the user to securely accept the payment, and to perform the cryptographic operations in a context that is protected from the Web shop. In particular, it is secure against a merchant that generates a page that looks like the payment page from the wallet (Figure 8b), as such a page would still not have access to the private keys of the coins that are in the wallet. If the user accepts, the resulting signed coins are transferred from the client to the server, again by a protocol that the merchant can customize to fit the existing infrastructure.

Instead of adding any cryptographic logic to the merchant frontend, the generic Taler merchant backend allows the implementor to delegate handling of the coins to the payment backend, which validates the coins, deposits them at the exchange, and finally validates and persists the receipt from the exchange. The merchant backend then communicates the result of the transaction to the frontend, which is then responsible for executing the business logic to fulfill the order. As a result of this setup, the cryptographic details of the Taler protocol do not have to be re-implemented by each merchant. Instead, existing Web shops implemented in a multitude of programming languages can rather trivially add support for Taler by **(0)** detecting in the browser that Taler is available, **(1)** upon request, generating a contract in JSON based on the shopping cart, **(2)** allowing the backend to sign the contract before sending it to the client, **(7)** passing coins received in payment for a contract to the backend and **(8)** executing fulfillment business logic if the backend confirms the validity of the payment.

To setup a Taler backend, the merchant only needs to let it know his wire transfer routing details, such as an IBAN number. Ideally, the merchant might also want to obtain a certificate for the public key generated by the backend for improved authentication. Otherwise, the customer's authentication of the Web shop simply continues to rely upon HTTPS/X.509.

4. DISCUSSION

We will now discuss how customer's may experience relevant operational risks and failure modes of Taler, and relate them to failure modes in existing systems.

4.1 Security risks

In Taler, customers incur the risk of wallet loss or theft. We believe customers can manage this risk effectively because they manage similar risks of losing cash in a physical wallet. Unlike physical wallets, Taler's wallet could be backed up to secure against loss of a device.

Taler's contracts do provide a degree of protection for customers because they are signed by the merchant and retained by the wallet: while they mirror the paper receipts that customers may receive in physical stores, Taler's cryptographically signed contracts ought to carry more weight in courts than typical paper receipts.

Point-of-sale systems providing printed receipts have been compromised in the past by merchants to embezzle sales taxes. [**?**] With Taler, the merchant still generates a receipt for the customer; however, the record for the tax authorities ultimately is anchored with the exchange's wire transfer to the merchant. Using the subject of the wire transfer, the state can trace the payments and request the merchant to provide cryptographically matching contracts. Thus, this type of tax fraud is no longer possible, which is why we call Taler *taxable*. The mere threat of the state sometimes tracing transactions and contracts back to the merchant also makes Taler unsuitable for illegal activities.

The exchange operator is obviously crucial for risk management in Taler, as the exchange operator holds the customer's funds in a reserve in escrow until the respective deposit request arrives[4]. To ensure that the exchange operator does not embezzle these funds, Taler expects exchange operators to be regularly audited by an independent auditor[5]. The auditor can then verify that the incoming and outgoing transactions and the current balance of the exchange match the logs with the cryptographically secured transaction records.

4.2 Failure modes

There are several failure modes the user of a Taler wallet may encounter:

- As Taler supports multiple exchanges, there is a chance that a merchant might not support any exchange where the customer maintains a wallet balance. We mitigate this problem by allowing merchants to support all exchanges audited by a particular auditor. We believe this a reasonable approach, because auditors and merchants must operate with a particular legal and financial framework anyways. We note that a similar failure mode exists with credit cards, where not all merchants accept all issuers, especially internationally.
- Restoring the Taler wallet state from previous backups, or copying the wallet state to a new machine, may cause honest users to attempt to double spend coins, as the wallet does not know when coins are spent between backup and recovery. In this case, the exchange provides cryptographic proof that the coins were previously spent, so the wallet can verify that the exchange and merchant are behaving honestly. However, this gives rise to another subsequent failure mode, namely that ...
- There could be insufficient funds in the Taler wallet when making a payment. Usually the wallet can trivially check this before beginning a transaction, but when double-spending is detected this may also happen after the wallet already initiated the payment. This would usually only happen if the wallet is unaware of a backup operation voiding its internal invariants. If a payment fails in-flight due to insufficient funds, the wallet can use Taler's refresh protocol to obtain a refund for those coins that were not actually double-spent, and then explain the user that the balance was inaccurate due to inconsistencies from recovery, and overall insufficient for payment. For the user, this failure mode appears equivalent to an insufficient balance or credit line when paying with cards.

4.3 Comparison

The different payment systems discussed make use of different security technologies, which has an impact on their usability and the assurances they can provide. Except for Bitcoin, all payment systems described involve an authentication step. With Taler, the authentication itself is straightforward, as the customer is at the time visiting the Web portal of the bank, and the authentication is with the bank (Figure 5). With PayPal, the shop redirects the customer to the PayPal portal (step 5 in Figure 3) after the user selected PayPal as the payment method. The customer then provides the proof of payment to the merchant. Again, this is reasonably natural. The 3DS workflow (Figure 1) has to deal with a multitude of banks and their different implementations, and not just a single provider. Hence, the interactions are more complicated as the merchant needs to additionally perform a lookup in the card scheme directory and verify availability of the bank (steps 8 to 12).

The key difference between Taler and 3DS or PayPal is that authentication is done ahead of time, not at the time of purchase. After authenticating once to withdraw digital coins, the customer can perform many micropayments without having to reauthenticate. While this simplifies the process of the individual purchase, it shifts the mental overhead to an earlier time, and thus requires some planning, especially given that the digital wallet is likely to only contain a small fraction of the customer's available funds. As a result, Taler improves usability if the customer is able to withdraw funds once to then facilitate many micropayments, while Taler is likely less usable if for each transaction the customer first visits the bank to withdraw funds. This is deliberate, as Taler can only achieve reasonable privacy for customers if they do keep a balance in their wallet, thereby breaking the association between withdrawal and deposit.

Bitcoin's payment process (Figure 2) resembles that of Taler in one interesting point, namely that the wallet is given details about the contract the user is to en-

[4]As previoulsy said, this *deposit request* is aimed to transalte *coins* into real money and it's accomplished by a merchant after successfully receiving coins by a wallet. In other words, it is the way merchants get real money on their bank accounts

[5]Auditors are typically run by states

ter (steps 7 to 11). However, in contrast to Taler, here the Bitcoin wallet(s) are expected to fetch the "invoice" from the merchant, while in Taler the browser provides the Taler wallet with the proposed contract directly. In PayPal and 3DS, the user is left without a cryptographically secured receipt.

Card-based payments (including 3DS) and PayPal also extensively rely on TLS for security. The customer is expected to verify that his connections to the various Web sites are properly authenticated using X.509, and to know that it is fine to providing his bank account credentials to the legitimate `verifiedbyvisa.com`.[6] However, relying on users understanding their browser's indications of the security context is inherently problematic. Taler addresses this challenge by ensuring that digital coins are only accessible from fully wallet-generated pages, hence there is no risk of Web pages mimicking the look of the respective page, as they would still not obtain access to the digital coins.

Once the payment process nears its completion, merchants need to have some assurance that the contract is now valid. In Taler, merchants obtain a non-repudiable confirmation of the payment. With 3DS and PayPal, the confirmation may be disputed later (i.e. in case of fraud), or accounts may be frozen arbitrarily [**?**]. Payments in cash require the merchant to assume the risk of receiving counterfeit money. Furthermore, merchants have the cost maintaining change and depositing the money earned. With Bitcoin, there is no definitive time until a payment can be said to be confirmed (step 19, Figure 2), leaving merchants in a bit of a tricky situation.

5. CONCLUSION

Customers and merchants should be able to easily adapt their existing mental models and technical infrastructure to Taler. In contrast, Bitcoin's payment models fail to match common expectations, be it in terms of performance, durability, security, or privacy. Minimizing the need to authenticate to pay fundamentally improves usability.

We expect that electronic wallets that automatically collect digitally signed receipts for transactions will become commonplace. A key question for the future is thus whether this data collection will be done on behalf of the citizens and under their control, or on behalf of the Reich of big data corporations.

We encourage readers to try our prototype for Taler at `https://demo.taler.net/`, and to ponder why the billion dollar e-commerce industry still relies mostly on TLS for security, given that usability, security and privacy can clearly *all* be improved simultaneously using a modern payment protocol.

Acknowledgements

Removed for anonymous submission.

[6]The search query "verifiedbyvisa.com legit" is so common that, when we entered "verifiedbyvisa" into a search engine, it was the suggested autocompletion.

```
function handleInstall() {
  var show = document.getElementsByClassName("taler-installed-show");
  var hide = document.getElementsByClassName("taler-installed-hide");
  for (var i = 0; i < show.length; i++) {
    show[i].style.display = "";
  }
  for (var i = 0; i < hide.length; i++) {
    hide[i].style.display = "none";
  }
};

function handleUninstall() {
  var show = document.getElementsByClassName("taler-installed-show");
  var hide = document.getElementsByClassName("taler-installed-hide");
  for (var i = 0; i < show.length; i++) {
    show[i].style.display = "none";
  }
  for (var i = 0; i < hide.length; i++) {
    hide[i].style.display = "";
  }
};

function probeTaler() {
  var eve = new Event("taler-probe");
  console.log("probing␣taler");
  document.dispatchEvent(eve);
};

function initTaler() {
  handleUninstall(); probeTaler();
};

document.addEventListener("taler-wallet-present", handleInstall, false);
document.addEventListener("taler-unload", handleUninstall, false);
document.addEventListener("taler-load", handleInstall, false);
window.addEventListener("load", initTaler, false);
```

Figure 9: Sample code to detect the Taler wallet. Allowing the Web site to detect the presence of the wallet leaks one bit of information about the user. The above logic also works if the wallet is installed while the page is open.

```
/* Trigger Taler contract generation on the server, and pass the
   contract to the extension once we got it. */
function taler_pay(form) {
  var contract_request = new XMLHttpRequest();

  /* Note that the URL we give here is simply an example
     and not dictated by the protocol: each web shop can
     have its own way of generating and transmitting the
     contract, there just must be a way to get the contract
     and to pass it to the wallet when the user selects 'Pay'. */
  contract_request.open("GET", "generate-taler-contract", true);
  contract_request.onload = function (e) {
    if (contract_request.readyState == 4) {
      if (contract_request.status == 200) {
        /* Send contract to the extension. */
        handle_contract(contract_request.responseText);
      } else {
        /* There was an error obtaining the contract from the merchant,
           obviously this should not happen. To keep it simple, we just
           alert the user to the error. */
        alert("Failure␣to␣download␣contract␣" +
              "(" + contract_request.status + "):\n" +
              contract_request.responseText);
      }
    }
  };
  contract_request.onerror = function (e) {
    /* There was an error obtaining the contract from the merchant,
       obviously this should not happen. To keep it simple, we just
       alert the user to the error. */
      alert("Failure␣requesting␣the␣contract:\n" +
            contract_request.statusText);
  };
  contract_request.send();
}
```

Figure 10: Sample code to pass a contract to the Taler wallet. Here, the contract is fetched on-demand from the server. The `taler_pay()` function needs to be invoked when the user triggers the checkout.

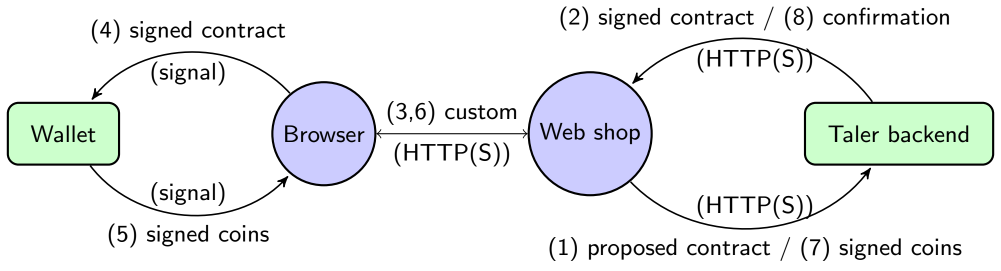


Figure 11: Both the customer's client and the merchant's server execute sensitive cryptographic operations in a secured background/backend that is protected against direct access. Interactions with the Taler exchange from the wallet background to withdraw coins and the Taler backend (Figure 4) to deposit coins are not shown. Existing system security mechanisms are used to isolate the cryptographic components (boxes) from the complex rendering logic (circles), hence the communication is restricted to JavaScript signals or HTTP(S) respectively.