Enabling Secure Web Payments with GNU Taler

Jeffrey Burdges, Florian Dold, Christian Grothoff, and Marcello Stanisci

Inria Rennes - Bretagne Atlantique
FIRSTNAME.LASTNAME@inria.fr

Abstract. GNU Taler is a new electronic online payment system which provides privacy for customers and accountability for merchants. It uses an exchange service to issue digital coins using blind signatures, and is thus not subject to the performance issues that plague Byzantine fault-tolerant consensus-based solutions.
The focus of this paper is addressing the challenges payment systems face in the context of the Web. We discuss how to address Web-specific challenges, such as handling bookmarks and sharing of links, as well as supporting users that have disabled JavaScript. Web payment systems must also navigate various constraints imposed by modern Web browser security architecture, such as same-origin policies and the separation between browser extensions and Web pages. While our analysis focuses on how Taler operates within the security infrastructure provided by the modern Web, the results partially generalize to other payment systems.
We also include the perspective of merchants, as existing systems have often struggled with securing payment information at the merchant's side. Here, challenges include avoiding database transactions for customers that do not actually go through with the purchase, as well as cleanly separating security-critical functions of the payment system from the rest of the Web service.

1 Introduction

The Internet needs a secure, usable and privacy-preserving micropayment system, which 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 [10]. Controlling money supply is critical to ensure stable prices that facilitate trade [40] instead of speculation [25].

Internet transactions, such as sending an e-mail or reading a Web site, tend to be of smaller commercial value than traditional transactions involving the exchange of physical goods. Consequently, if we want to associate payments with these types of transactions, we face the challenge of reducing the mental and technical overheads of existing payment systems. For example, executing a 3-D Secure [28] payment process (Figure 1) takes too long, is way too complex, and way too expensive to be used for payment for typical Web articles.

Addressing this problem is urgent: ad-blocking technology is eroding advertising as a substitute for micropayments [37], and the Big Data business model in which citizens pay with their private information [12] in combination with the deep state hastens our society's regression towards post-democracy [36].

The focus of this paper is GNU Taler, a new free software payment system designed to meet certain key ethical considerations from a social liberalism perspective. In Taler, the paying customer remains anonymous while the merchant is easily identified and thus taxable. Here, *anonymous* simply means that the payment process 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 for any transaction the state can easily obtain the necessary information about the identity of the merchant and the respective contract in order to levy income, sales, or value-added taxes. Taler uses blind signatures [6] to create digital coins and a new *refresh* protocol [9] to allow giving change and refunds while maintaining unlinkability.

This paper will not consider the details of Taler's cryptographic protocols.[1] The basic cryptography behind blind-signature based payment systems has been known for over 25 years [7]. However, it was not until

[1] Details of the protocol are documented at `https://api.taler.net/`

2015 that the W3C started the payments working group [41] to explore requirements for deploying payment systems that are more secure and easy to use for the Web. Our work describes how a modern payment system using blind signatures could practically be integrated with the modern Web to improve usability, security, and privacy. This includes the challenge of hiding the cryptography from the users, integrating with modern browsers, integrating with Web shops, providing proper cryptographic proofs for all operations, and handling network failures. We explain our design using terms from existing *mental models* that users have from widespread payment systems.

Key contributions of this paper are:

- A description of different payment systems using common terminology, which allows us to analytically compare these systems.
- 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 browsers.
- A publicly available free software reference implementation of the presented architecture.

2 Existing payment workflows

Before we look at the payment workflow for Taler, we sketch the workflow of existing payment systems. This establishes a common terminology which we will use to compare different payment processes. We include interaction diagrams for some of the payment systems based on resources from the W3C payment interest group.

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 both buyer and seller roles, just 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, in turn, *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 [14], 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 in theory by serial numbers printed on bills [23], which make each bill *unique*.

Spending cash does not provide any inherent *proof of purchase* for the customer. Instead, the merchant provides 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 risk to the amount of cash that they carry or accept at a given time [21]. Additionally, customers are advised to choose the ATMs they use carefully, as malicious ATMs may attempt to *steal* their customer's credentials [13]. Authentication with an ATM can involve a special ATM card, or the use of credit or debit cards. In all these cases, these physical security tokens are issued by the customer's bank.

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. Secure systems typically combine multiple forms of authentication including secret information, such as personal identification

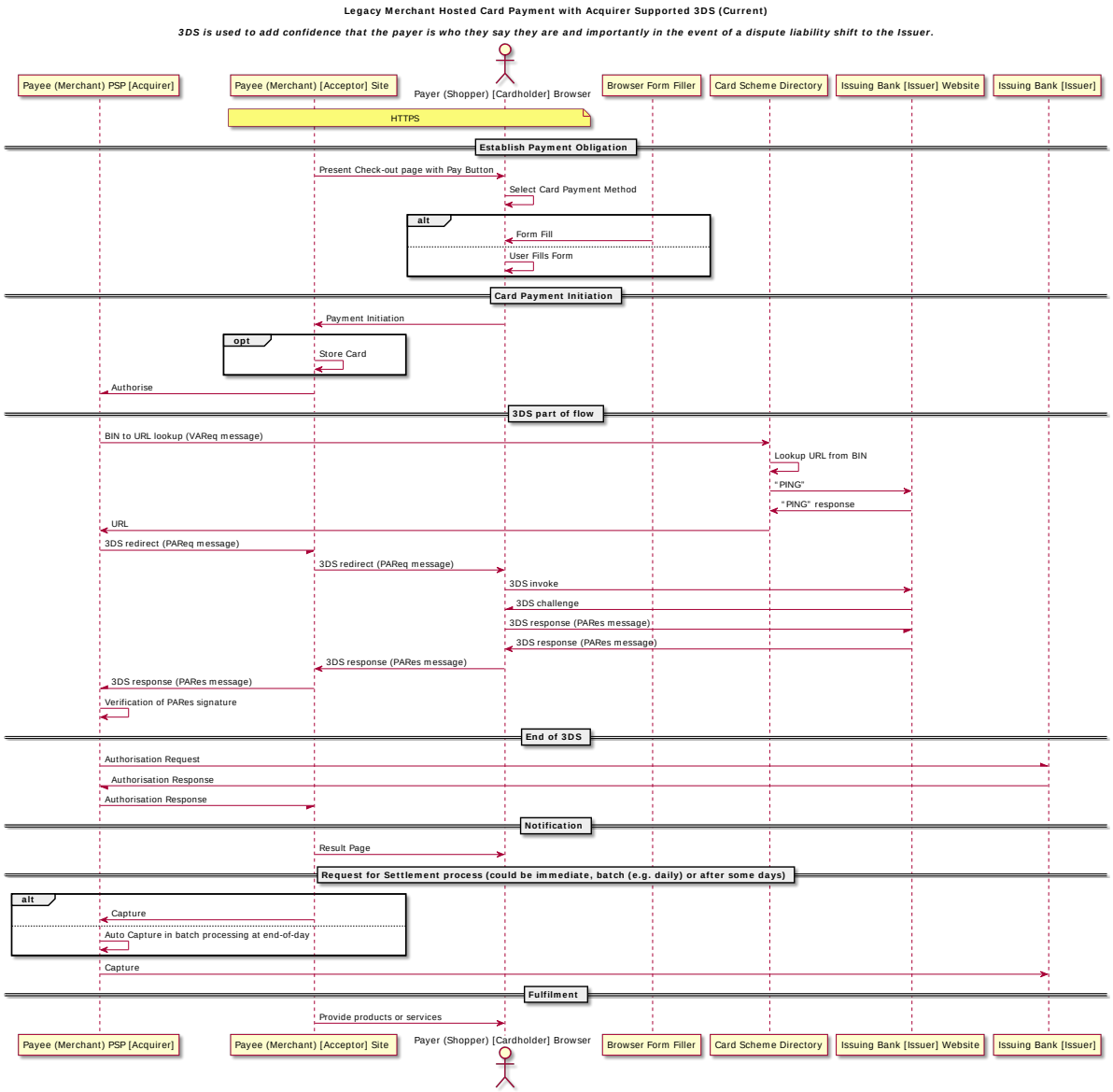


Fig. 1. Card payment processing with 3-D Secure (3DS). (From: W3C Web Payments IG.)

numbers (PINs), transaction numbers (TANs) [1] or credit card verification (CCV) codes, and physical security devices such cards with an EMV chip [2], TAN generators, or the customer's mobile phone [11]. A typical modern Web payment process involves: (1.) the merchant offering a secure communication channel using TLS based on the X.509 public key infrastructure;[2] (2.) selecting a *payment method*; (3.) entering the credit card details like the owner's name, card number, expiration time, CCV code, and billing address; and (4.) (optionally) authorizing the transaction via mobile TAN, or by authenticating against the customer's bank. Due to the complexity of this, the data entry is often performed on a Web site that is operated by a third-party payment processor and *not* the merchant or the customer's bank. Figure 1 shows a typical card-based payment process on the Web using the UML style of the W3C payment interest group [41]. Most

[2] Given numerous TLS protocol and implementation flaws as well as X.509 key management incidents in recent years [18], one cannot generally assume that the security provided by TLS is adequate under all circumstances.

of the details are not relevant to this paper, but the diagram nicely illustrates the complexity of the common 3-D secure process.

Given this process, there is an inherent risk of information leakage of customers' credentials. *Fraud detection* systems attempt to detect misuse of stolen credentials, and payment system providers handle disputes between customers and merchants. As a result, Web payment processes may finish with (5.) the payment being rejected for a variety of reasons, such as false positives in fraud detection or 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 to shift dispute *liability* to the issuer of the card—who may then try to shift it to the customer [28, §2.4]. Even in cases where the issuer or the merchant remain legally first in line for liabilities, there are still risks customers incur from the card dispute procedures, such as neither them nor the payment processor noticing fraudulent transactions, or them noticing fraudulent transactions past the *deadline* until which their bank would reimburse 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 cryptographically *verifiable* electronic receipts for the customer, which theoretically enables malicious merchants to later change the terms of the contract.

Beyond these primary 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 procedure of *notifying* the bank in the first place. As a result, customers must remain wary about using their cards, which limits their online shopping [33, p. 50].

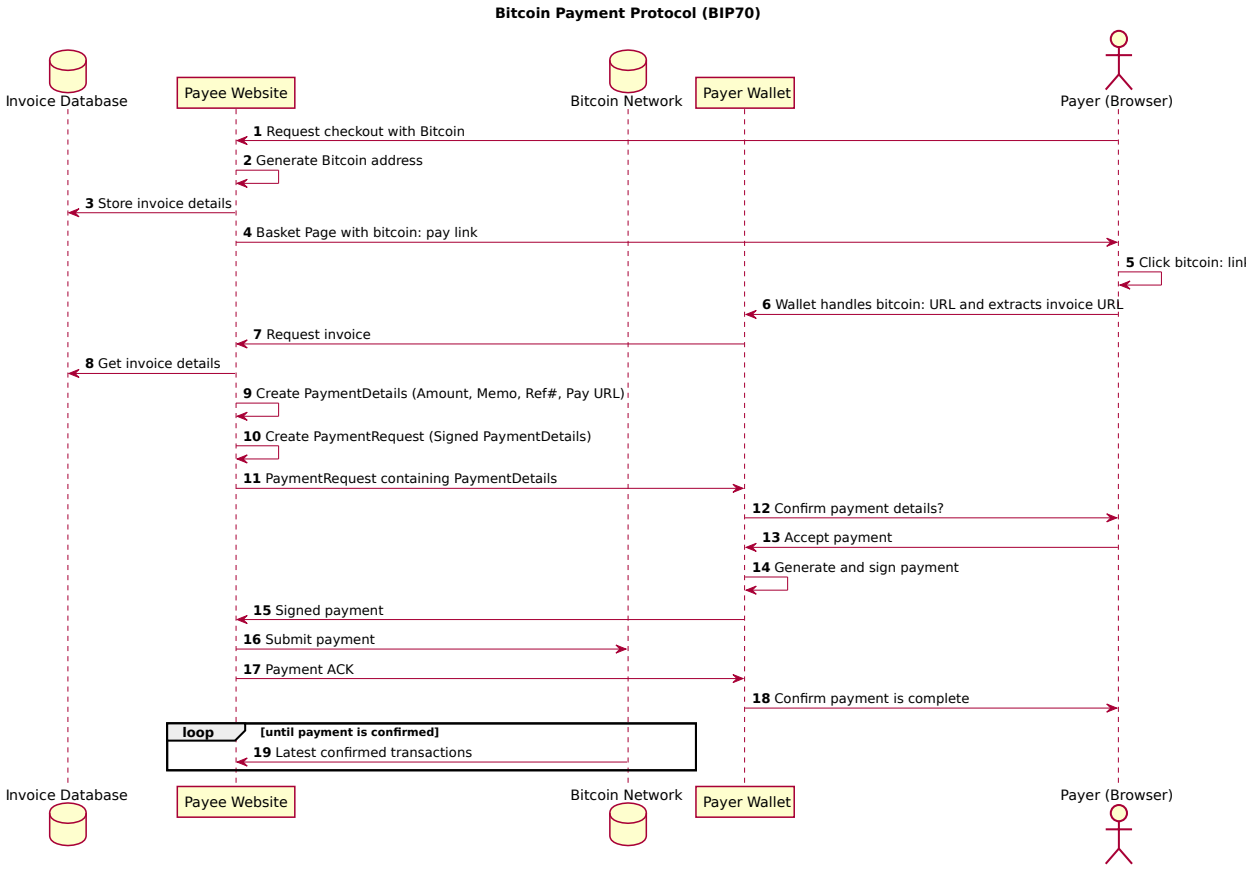


Fig. 2. Bitcoin payment processing. (From: W3C Web Payments IG.)

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 must know the corresponding private key 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 denominated in a new currency labeled BTC, whose valuation depends upon *speculation*, as there is no authority that could act to stabilize exchange rates or force anyone to accept BTC as *legal tender* to settle obligations. 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, which is called *mining*.

The mining process is incentivised by a combination of transaction fees and mining rewards [29]. The latter process also provides primitive accumulation [31] for BTC. Conversion to BTC from other currencies and vice versa incurs substantial fees [5]. 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 [39].

The classical Bitcoin payment workflow consisted of entering payment details into a peer-to-peer application. The user would access their Bitcoin *wallet* and instruct it to transfer a particular amount from one of his accounts to the account of the merchant. He could possibly include additional metadata to be associated with the transfer to be 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 payments significantly less browser-friendly than ordinary card payments, as illustrated in Figure 2. This has led to the development of browser-based wallets.[3]

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, it is recommended to wait for 6 blocks (on average one hour) before considering a transaction committed [29]. In cases where merchants are unable to accommodate this delay, they incur significant fraud risks.

Bitcoin is considered to be secure against an adversary who cannot control around a fifth of the Bitcoin miner's computational resources [3, 15, 17]. As a result, the network must expend considerable computational resources to keep this value high. According to [26], a single Bitcoin transaction uses roughly enough electricity to power 1.57 American households for a day. These costs are largely hidden by speculation in BTC, but that speculation itself contributes to BTC's valuation being volatile. [19, 24, 25].

Bitcoin's pseudononymity applies equally to both customers and merchants, which makes Bitcoin amenable to tax evasion, money laundering, sales of contraband, and especially extorion [30]. As a result, anonymity tools like mixnets do not enjoy widespread support in the Bitcoin community where many participants seek to make the currency appear more legitimate. While Bitcoin's transactions are difficult to track, there are several examples of Bitcoin's pseudononymity being broken by investigators [32]. This has resulted in the development of new protocols with better privacy protections.

Zerocoin [27] is such an extension of Bitcoin: It affords protection against linkability of transactions, but at non-trivial additional computational costs even for spending coins. This currently makes using Zerocoin unattractive for payments, especially with mobile devices.

Bitcoin also faces serious scalability limitations, with the classic implementation being limited to at most 7 transactions per second globally on average.[4] There are a variety of efforts to confront Bitcoin's scaling problems with off-blockchain techniques, like side-chains. Amongst these, the blind off-chain lightweight transactions (BOLT) proposal [16] provides anonymity by routing off-blockchain transfers through bank-like intermediaries. Although interesting, there are numerous seemingly fragile aspects of the BOLT protocol, including aborts deanonymizing customers, intermediaries risking unlimited losses, and theft if a party fails to post a refute message in a timely fashion.

[3] `https://github.com/frozeman/bitcoin-browser-wallet`
[4] `http://hackingdistributed.com/2016/08/04/byzcoin/`

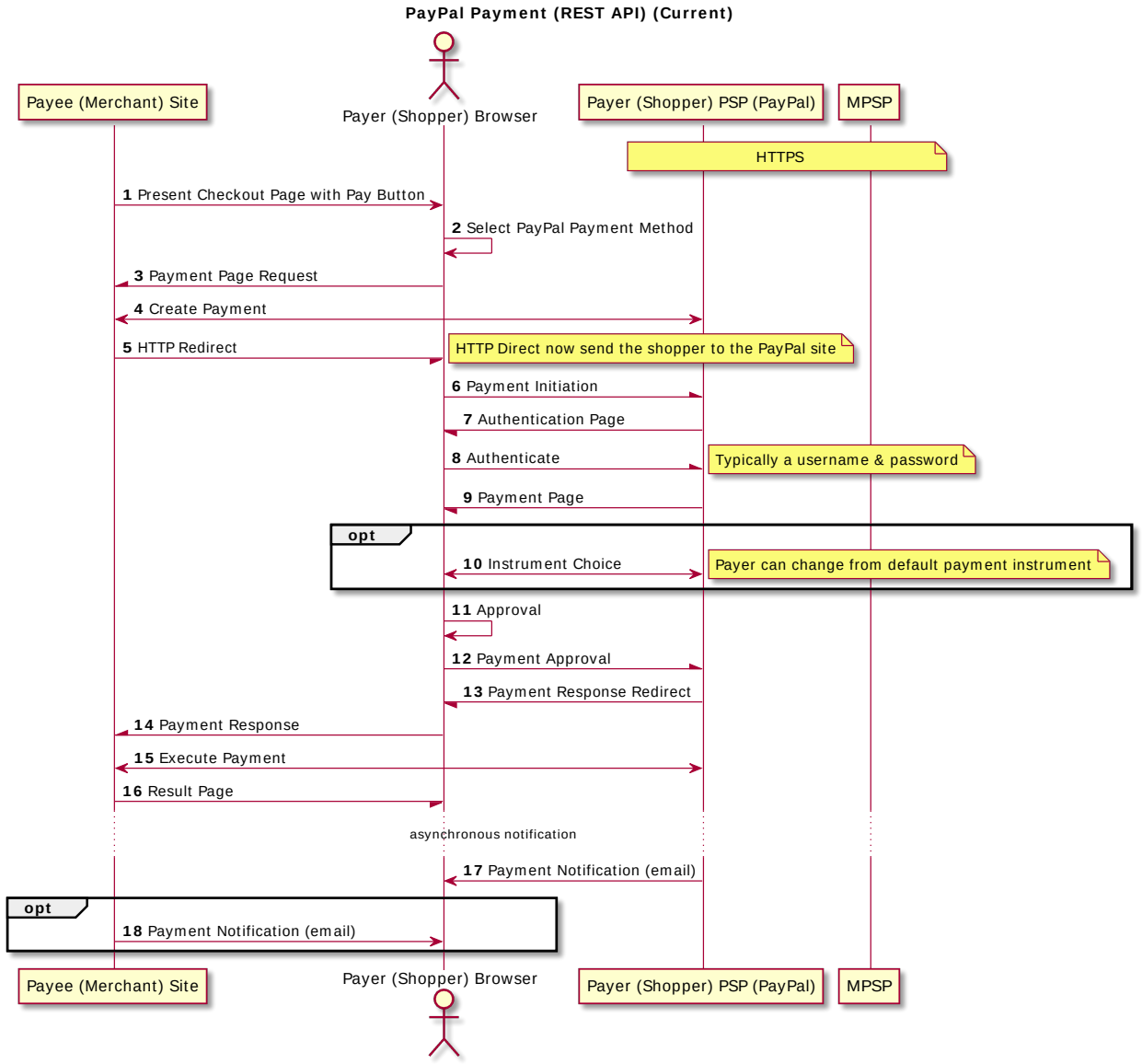


Fig. 3. Payment processing with PayPal. (From: W3C Web Payments IG.)

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 (see 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 customers and merchants 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 payment systems, these oligopolies are politically dangerous [35], and the lack of *competition* can result in excessive profit taking that may require political solutions [20] to the resulting *market failure*.

The use of 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. It has an open protocol specification, which couples cash-like anonymity for customers 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* [6]. Since their discovery thirty years ago, cryptographers have viewed blind signatures as the optimal cryptographic primitive for privacy-preserving consumer-level transaction systems. However, previous transaction systems based on blind signatures have failed to see widespread adoption. This paper details strategies for hiding the complexity of the cryptography from users and integrating smoothly with the Web, thereby providing crucial steps to bridge the gap between good cryptography and real-world deployment.

There are four key roles in the Taler system (Figure 4):

- *Customers* use a digital wallet to withdraw, hold, and spend coins. Wallets manage the customer's accounts at the exchange, and keep receipts in a transaction history. Wallets can be realized as browser extensions, mobile Apps or even in custom hardware. If a user's digital wallet is compromised, the current balance may be lost, just as with an ordinary wallet containing cash. A wallet includes a list of trusted auditors, and will warn users against using an exchange that is not certified by a trusted auditor.
- *Exchanges*, which are run by financial service providers, enable customers to withdraw anonymous digital coins, and merchants to deposit digital coins, in exchange for bank money. Coins are signed by the exchange using a blind signature scheme [6]. Thus, only an exchange can issue new coins, but coins cannot be traced back to the customer who withdrew them. Furthermore, 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 used by the customer in return for a bank wire transfer of their value. While the exchange is determined by the customer, the merchant's contract specifies the currency, a list of accepted auditors, and the maximum exchange deposit fee the merchant is willing to pay. Merchants consist of a *frontend*, which interacts with the customer's wallet, and a *backend*, which 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. Auditors are typically operated by or on behalf of financial regulatory authorities. Depending on local legislation, auditors may mandate that exchanges have enough financial reserves before authorizing them to create a given volume of signed digital coins to provide a buffer against potential risks due to operational failures (such as data loss or theft of private keys) of the exchange. Auditors certify exchanges that they audit using digital signatures. The respective signing keys of the auditors are distributed to customer and merchants.

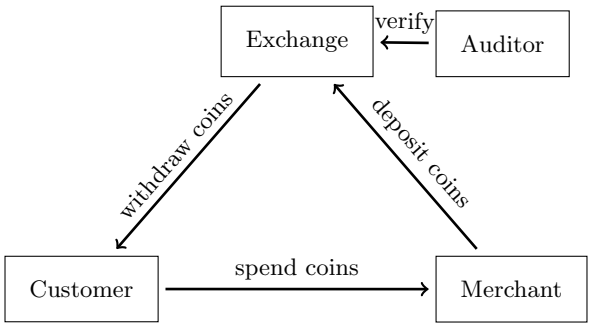


Fig. 4. Taler system overview.

The specific protocol between wallet and merchant depends on the setting. For a traditional store, a near field communication (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 and explain how the actors in the Taler system interact by way of a typical payment.

Initially, the customer installs the Taler wallet extension for their browser. This only needs to be done once per browser. Naturally, this step may become superfluous if Taler is integrated tightly with browsers in the future. Regardless, installing the extension involves only one or two clicks to confirm the operation. Restarting the browser is not required.

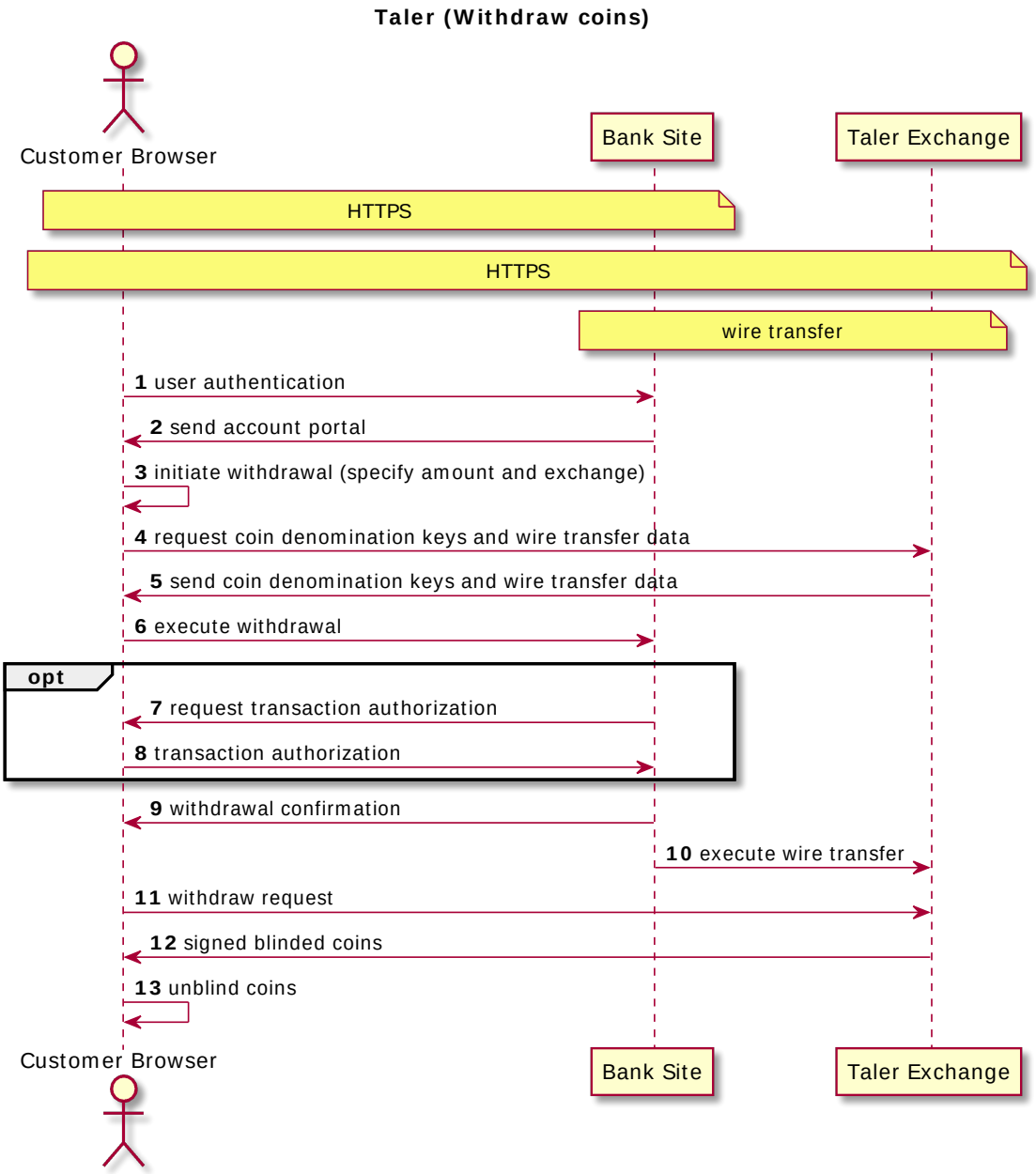


Fig. 5. Withdrawing coins with Taler.

3.1 Withdrawing coins

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

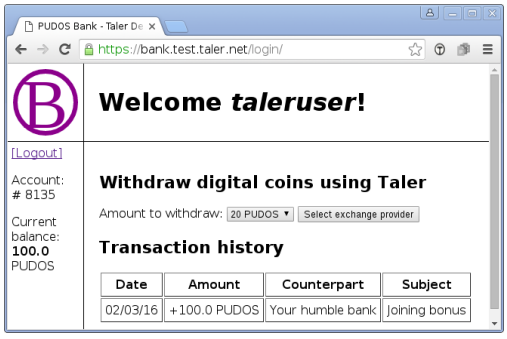
The next step depends on the level of Taler support offered by the bank:

- If the bank does not offer integration with Taler, the customer needs to use the menu of the wallet to create a *reserve*. The wallet will ask which amount in which *currency* (e.g. 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, the above is exactly the kind of interaction we would like to avoid for usability reasons.
- Otherwise, if the bank fully supports Taler, the customer has a form in the online banking portal in which they 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.

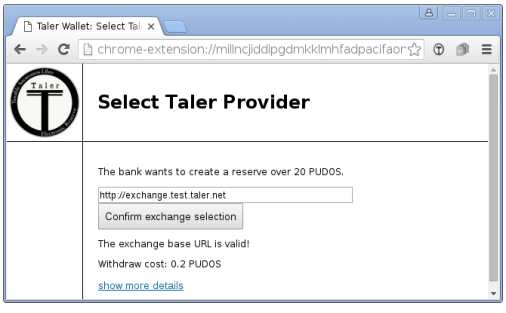
In either case, the wallet can then withdraw the coins from the exchange, and does so in the background without further interaction with the customer.

In principle, the exchange can be directly operated by the bank, in which case the step where the customer selects an exchange could 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.

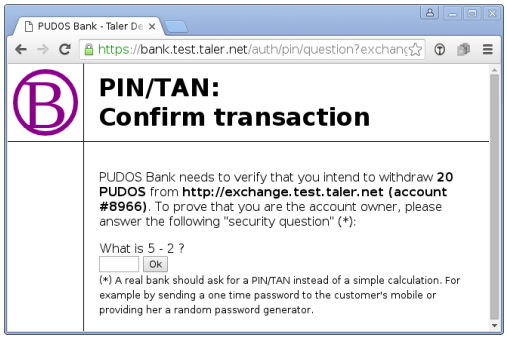
(a) Bank login. (Simplified for demonstration.)



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



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



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

Fig. 6. Required steps in a Taler withdrawal process.

3.2 Spending coins

At a later point in time, the customer can spend their coins by visiting a merchant that accepts digital coins in the respective currency 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 do not matter for the customer, as we expect most merchants to accept most exchange providers accredited by the auditors that wallets include by default. Similarly, we expect exchanges to operate with transaction fees acceptable to most merchants to avoid giving customers a reason to switch to another exchange. If transaction fees are higher than what is covered by the merchant, the customer may choose to cover them.

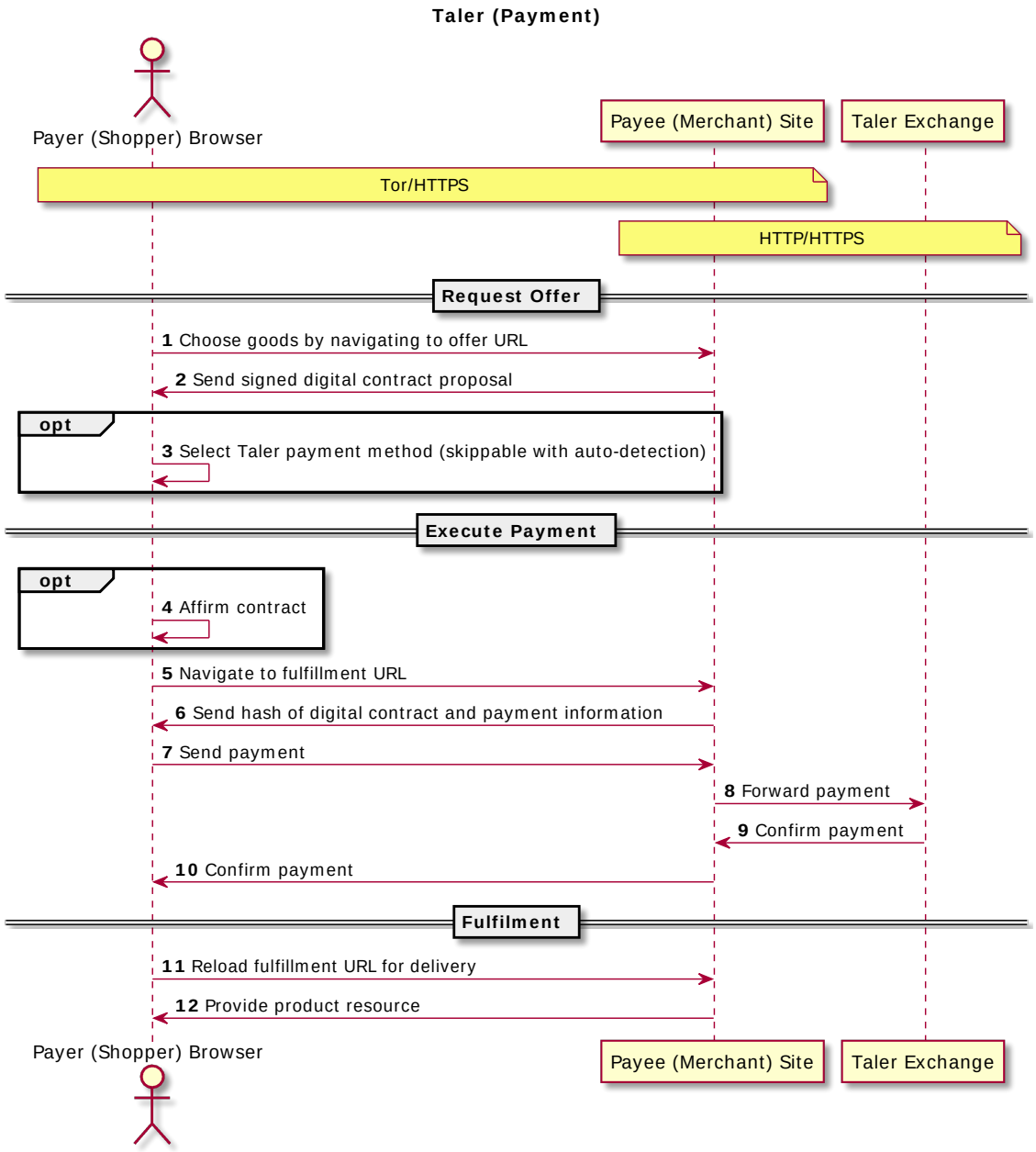
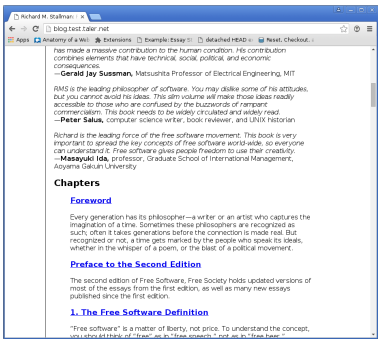
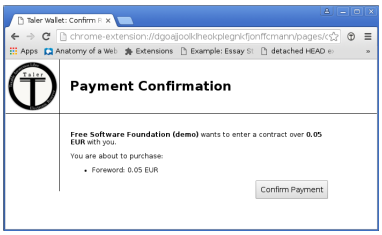


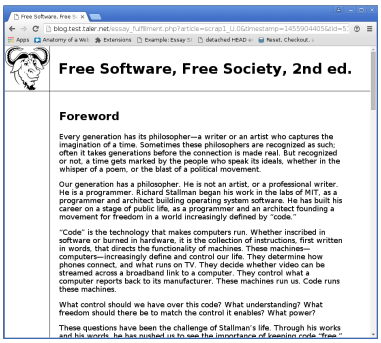
Fig. 7. Payment processing with Taler.



(a) Select article.
Generated by Web shop.



(b) Confirm payment.
Generated by Taler wallet.



(c) Receive article.
Generated by Web shop.

Fig. 8. Required steps in a Taler checkout process.

```
<script src="taler-wallet-lib.js"></script>
<script>
  taler.onPresent(() => {
    alert("Taler wallet is installed");
  });
  taler.onAbsent(() => {
    alert("Taler wallet is not installed");
  });
</script>
```

Fig. 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.

```
HTTP/1.1 402 Payment Required
Content-Type: text/html; charset=UTF-8
X-Taler-Contract-Url: https://shop/generate-contract/42

<!DOCTYPE html>
<html>
  <!-- fallback for browsers without the Taler extension -->
  You do not seem to have Taler installed, here are other payment options ...
</html>
```

Fig. 10. Sample HTTP response to prompt the wallet to show an offer.

```
<script src="taler-wallet-lib.js"></script>
<script>
  taler.offerContractFrom("https://shop/generate-contract/42", (err) => {
    alert("Error while offering contract");
  });
</script>
```

Fig. 11. Sample JavaScript code to prompt the wallet to show an offer. Here, the contract is fetched on-demand from the server. The `taler_pay()` function needs to be invoked when the user triggers the checkout.

```
{
  "H_wire":"YTH0C4QBCQ10VDNTJN0DCTTV2Z6JHT5NF43F0RQHZ8JYB5NG4W4G...",
  "amount":{"currency":"EUR","fraction":1,"value":0},
  "auditors":[{"auditor_pub":"42V6TH91Q83FB846DK1GW3JQ5E8DS273W4236AXC397892ESD0B0"}],
  "exchanges":[{"master_pub":"1T5FA8VQHMMKBHDMYPRZA2ZFK2S63AKF0YTHJZWFKF45K2JGC8H0",
                "url":"https://exchange/"}],
  "expiry":"/Date(1480119270)/",
  "fulfillment_url": "https://shop/article/42?tid=249960194066269&time=1471479270",
  "max_fee":{"currency":"EUR","fraction":01,"value":0},
  "merchant":{"address":"Mailbox␣4242","jurisdiction":"Jersey","name":"Shop␣Inc."},
  "merchant_pub":"Y1ZAR5346J3ZTEXJCHQY9NJN78EZ2HSKZK8M0MYTNRJG5N0HD520",
  "products":[{
    "description":"Essay:␣The␣GNU␣Project",
    "price":{"currency":"EUR","fraction":1,"value":0},
    "product_id":42,"quantity":1}],
  "refund_deadline":"/Date(1471522470)/",
  "timestamp":"/Date(1471479270)/",
  "transaction_id":249960194066269
}
```


Fig. 12. Minimal Taler contract over a digital article with a value of €0.10. The merchant will pay transaction fees up to €0.01. The hash over the wire transfer information was truncated to make it fit to the page.

As with traditional Web transactions, customers first select which items they wish to buy. This can involve building a traditional shopping cart, or simply clicking on a particular link for the respective article (Figure 8a). Once the articles have been selected, the Web shop directs the user to the *offer* URL, where the payment details are negotiated. The process usually starts by allowing the user to select a *payment method* from a set of methods supported by the Web shop. Taler also allows the Web shop to detect the presence of a Taler wallet (Figure 9), so that the selection of alternative payment methods can be skipped if a Taler wallet is installed (as it is in Figure 8).

Offer The offer URL of the Web shop can then initiate payments by sending a *contract proposal* (Figure 12) to the wallet, either via the HTTP status code `402 Payment Required` (Figure 10), or via Taler's JavaScript API (Figure 11). 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 of the terms and prices. In case that transaction fees need to be covered by the customer, these are shown together with the rest of the proposed contract.

The Taler wallet operates from a securely isolated *background* context on the client side. The user interface that displays the contract and allows the user to confirm the payment is displayed by this background context. By running in the background context, the wallet can perform the cryptographic operations protected from the main process of the Web site. In particular, this architecture is secure against a merchant that generates a page that looks like the wallet's payment page (Figure 8b), as such a page would still not have access to the private keys of the coins that are exclusive to the background context.

If the customer approves the contract by clicking the "Confirm Payment" button (Figure 8b), their 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 the *fulfillment* URL provided by the merchant in the contract (Figure 8c).

Fulfillment The fulfillment URL uniquely identifies a purchase by some customer, while the offer URL identifies a generic offer that is not specific to a customer. The purchase identified by a fulfillment URL may have been completed or still be in progress. The information contained in the fulfillment URL must allow the merchant to restore the full contract (including a unique transaction identifier) that was associated with the purchase, either directly from the URL or indirectly from an identifier in a database. Efficiently reconstructing

```
taler.executePayment("2BAH2AT4GSG5JRM2W4YWTSYGY66EK4X8CX2V69D5VF7XV703AJMG",
  "https://shop/pay", "https://shop/article/42",
  (err) => { alert("Sending payment failed"); });
```

Fig. 13. Sample JavaScript code to trigger transmission of a payment to the merchant.

```
HTTP/1.1 402 Payment Required
Content-Type: text/html; charset=UTF-8
X-Taler-Contract-Hash: 2BAH2AT4GSG5JRM2W4YWTSYGY66EK4X8CX2V69D5VF7XV703AJMG...
X-Taler-Pay-Url: https://shop/pay
X-Taler-Offer-Url: https://shop/article/42

<!DOCTYPE html>
<html>
  <!-- fallback for browsers without the Taler extension -->
  You do not seem to have Taler installed, here are other payment options ...
</html>
```

Fig. 14. Sample HTTP response when the user agent navigates to a fulfillment URL without the session state that indicates they have paid for the resource. Note that unlike in Listing 10, the response references a contract that typically is already known to the wallet via its hash code.

the contract entirely from the URL instead of using costly database transactions can be important, as costly disk operations for incomplete purchases make merchants more susceptible to denial-of-service attacks from adversaries pretending to be customers.

When a customer has completed a purchase, navigating to the fulfillment URL in a browser will show the resource associated with the purchase. This resource can be a digital good such as a news article, or simply a confirmation for products that are delivered by other means.

When a customer has not yet completed a purchase (this is always the case when a customer visits the fulfillment URL for the first time), or when the Web shop cannot confirm that this visitor has paid for the contract, for example because the session state was lost,[5] the Web store responds by (again) triggering a payment process (either via JavaScript or using `402 Payment Required`, see Figures 13 and 14). However, unlike the response from the offer URL, the 402 response from the fulfillment page includes the headers `X-Taler-Contract-Hash`, `X-Taler-Pay-Url` and `X-Taler-Offer-Url`.

If the contract hash matches a payment which the user already previously approved, the wallet reacts to this by injecting the logic to transmit the payment to the *pay* URL of the Web shop into the page. Then the wallet inspects the response as it may contain error reports about a failed payment which the wallet has to handle. By submitting the payment this way, we also ensure that this intermediate request does not require JavaScript and still does not interfer with navigation. Once the Web shop confirms the payment, the wallet causes the fulfillment URL to be reloaded.

If the contract hash does not match a payment which the user already approved, for example because the user obtained the link from another user, the wallet navigates to the offer URL included in the header.

Discussion Various failure modes are considered in this design:

- 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 the customer and the merchant, the wallet will try to recover control over the coins at the exchange by effectively spending the coins first using Taler's 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 via the transaction history kept

[5] This can happen when when privacy conscious users delete their cookies. Also, some user agents (such as the TOR browser) do not support persistent (non-session) cookies.

by the wallet, 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 (e.g. because it was restored from backup), updates the database and allows the user to retry the transaction.

While our design requires a few extra roundtrips, it has the following key advantages:

- It works in the confines of the WebExtensions API.
- It supports restoring session state for bookmarked Web resources even after the session state is lost by the user agent.
- Sending deep links to fullfilment or offer pages to other users has the expected behavior of asking the other user to pay for the resource.
- Asynchronously transmitting coins from injected JavaScript costs one roundtrip, but does not interfer with navigation and allows proper error handling.
- The different pages of the merchant have clear delineations: the shopping pages conclude by making an offer, and the fulfillment page begins with processing an accepted contract. It is thus possible for these pages to be managed by separate parties. The control of the fulfillment page over the transmission of the payment data minimizes the need for exceptions to handle cross-origin resource sharing [4, 22].
- The architecture supports security-conscious users that may have disabled JavaScript, as it is not necessary to execute JavaScript originating from Web pages to execute the payment process.

3.3 Giving change and refunds

An important cryptographic 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, the change giving process is completely hidden from the user. In fact, our 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 available 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 refresh protocol [9] also allows merchants to give refunds to customers. To refund a purchase, the merchant obtains a signed refund permission from the exchange, which the customer's wallet processes (Figure 15) to obtain new, unlinkable coins as refund. This process allows the customer to say anonymous when receiving refunds.

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

```
<script src="taler-wallet-lib.js"></script>
<script>
  // Obtain refund permissions from the merchant backend
  // ...
  let refundPermissions = /* ... */;
  taler.acceptRefunds(refundPermissions, (err) => {
    alert("An error occured while attempting a refund");
  });
</script>
```

Fig. 15. Sample JavaScript code to trigger a refund from the merchant's web shop

3.4 Deployment considerations for merchants

Payment system security is not only 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 (e.g. the card) and to guard a secret (e.g. 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 [34].

Taler simplifies the deployment of a secure payment system for merchants. The high-level cryptographic design provides the first major advantage, as merchants 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 [42]. In fact, the exchange does not need to have a formal business relationship with the merchant 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 direct 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.

The payment process requires a few cryptographic operations on the side of the merchant, such as signing a contract and verifying the customer's and the exchange's signatures. The merchant also needs to store transaction data, in particular so that the store can match sales with incoming wire transfers from the exchange. We simplify this for merchants by providing a generic payment processing *backend* for the Web shops.

Figure 16 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 customary Web shop pages, which are transmitted to the customer's browser. Once the order has been constructed, the shop frontend provides 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 shown in Figure 11).

Instead of adding any cryptographic logic to the merchant frontend, the Taler merchant backend allows the implementor to delegate coin handling 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 (Figure 16), 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 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

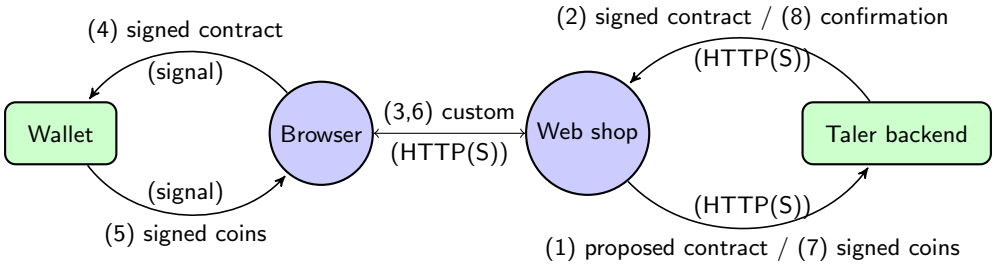


Fig. 16. Both the customer's client and the merchant's server execute sensitive cryptographic operations in a secured background/backend that is protected against direct access. Interactions with the Taler exchange from the wallet background to withdraw coins and the Taler backend to deposit coins are 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.

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 configure the wire transfer routing details, such as the merchant's IBAN number, as well as a list of acceptable auditors and limits for transaction fees. 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. We note that managing the risk does not imply that customers will never suffer from such a loss. We expect that customers will limit the balance they carry in their digital wallet. Ideally, the loss should be acceptable given that the customer gains the insight that their computer was compromised.

Taler's contracts 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 receive in physical stores, Taler's cryptographically signed contracts ought to carry more weight in courts than typical paper receipts. Customers can choose to discard the receipts, for example to avoid leaking their shopping history in case their computer is compromised.

Point-of-sale systems providing printed receipts have been compromised in the past by merchants to embezzle sales taxes. [38] 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 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[6] To ensure that the exchange operator does not embezzle these funds, Taler expects exchange operators to be regularly audited by an independent auditor[7]. The auditor can then verify that the incoming and outgoing transactions, and the current balance of the exchange matches the logs with the cryptographically secured transaction records.

4.2 Failure modes

There are several failure modes which a customer using 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 withdrew coins from. 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, which is often the case internationally.

[6] As previously said, this *deposit request* is aimed to exchange *coins* for bank money, and it is made by a merchant after successfully receiving coins from a wallet during the payment process.

[7] Auditors are typically run by financial regulatory bodies of states.

– 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 to the wallet that the coins were previously spent so the wallet can verify that the exchange and the merchant are behaving honestly.
– 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 invariant of knowing which coins have already been spent. 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 to the user that the balance was inaccurate due to inconsistencies, and insufficient for payment. For the user, this failure mode appears equivalent to an insufficient balance or credit line when paying with debit or credit cards.

In the future, we plan to make it easy for users to backup and synchronize wallets to reduce the probability of the later two failure modes. A key issue in this context is that these processes will need to be designed carefully to avoid leaking information that might allow adversaries to link purchases via side channels opened up by the synchronization protocol.

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 selects 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).

A key difference between Taler and 3DS or PayPal is that in Taler, authentication is done ahead of time. After authenticating once to withdraw digital coins, the customer can perform many micropayments without having to re-authenticate. 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 withdraws funds once to then perform 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 keep a balance in their wallet, as this is necessary to break 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 enters (steps 7 to 11). However, in contrast to Taler, Bitcoin wallets are expected to fetch the "invoice" from the merchant. In Taler, the browser can provide the proposed contract directly to the wallet. 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 their connections to the various Web sites are properly authenticated using X.509, and to know that it is fine to provide their bank account credentials to the legitimate `verifiedbyvisa.com`.[8] 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 wallet-generated pages. As such 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.

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

Once the payment process nears its completion, merchants need to have some assurance that the contract is valid. In Taler, merchants obtain a non-repudiable confirmation of the payment. With 3DS and PayPal, the confirmation may be disputed later (e.g. in case of fraud), or accounts may be frozen arbitrarily [8]. Payments in cash require the merchant to assume the risk of receiving counterfeit money. Furthermore, with cash merchants have the cost of maintaining change and depositing the money earned. The most extreme case for lack of assurances upon "completion" is Bitcoin, where there is no time until a payment can be said to be definitively confirmed (step 19, Figure 2), leaving merchants in a bit of a tricky situation.

Finally, attempts to address the scalability hudles of Bitcoin using side-chains or schemes like BOLT introduce semi-centralized intermediaries, not wholey unlike Taler's use of exchanges. Compared to BOLT, we would expect a Taler exchange operating in BTC to offer stronger security to all parties and stronger anonymity to customers, as well as being vastly cheaper to operate.

5 Future work

This paper has focused on how Taler would work for Web payments. However, the underlying cryptography should work just as well for other domains. In particular, we plan to adapt Taler for NFC and peer-to-peer payments in the future.

5.1 NFC payments

We have so far focused on how Taler could be used for Web payments; however, Taler can in theory also be 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 and confirm the payment on his device, which would then transfer the coins and obtain a receipt. A native NFC application would be less restricted in its interaction with the point-of-sale system compared to a browser extension, and the security of the communication channel is also comparable. Thus, running Taler over NFC is largely a simplification of the existing process.

In particular, there are no significant new concerns arising from an NFC device possibly losing contact with a point-of-sale system, as for Web payments, Taler already only employs 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.

5.2 Peer-to-peer payments

Peer-to-peer payments are in principle 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 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 straightforward because we presumably do not have to worry about man-in-the-middle attacks, 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, because after copying a private key both parties could try to spend the coins, but only the first transaction will succeed. 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 sender's exchange would have to support wire transfers to the receiver's exchange. 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. As such, in terms of institutions, one would expect this setup to be offered most easily by traditional banks, effectively merging the technical concepts of a (traditional) bank accounts and Taler reserves into one service for the customer.

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.

6 Conclusions

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 security and usability.

We expect that electronic wallets that automatically collect digitally signed receipts for transactions will become commonplace. By providing a free software wallet, Taler gives the user full control over the usage of their transaction history, as opposed to giving control to big data corporations.

We encourage readers to try our prototype for Taler at `https://demo.taler.net/`.

Acknowledgements

This work benefits from the financial support of the Brittany Region (ARED 9178) and a grant from the Renewable Freedom Foundation. We thank Bruno Haible for his financial support enabling us to participate with the W3c payment working group. We thank the W3C payment working group for insightful discussions about Web payments. We thank Krista Grothoff and Neal Walfield for comments on an earlier draft of the paper. We thank Gabor Toth for his help with the implementation.

References

1. Chiptan/cardtan: What you see is what you sign. `http://www.kobil.com/solutions/identity-access-card-readers/chiptan/` (2016)
2. Emvco. `http://www.emvco.com/` (2016)
3. Bahack, L.: Theoretical Bitcoin attacks with less than half of the computational power (draft). IACR Cryptology ePrint Archive 2013, 868 (2013), `http://eprint.iacr.org/2013/868`
4. Barth, A.: The Web Origin Concept. RFC 6454 (Proposed Standard) (Dec 2011), `http://www.ietf.org/rfc/rfc6454.txt`
5. Beigel, O.: What Bitcoin exchanges won't tell you about fees (2015), `https://www.cryptocoinsnews.com/what-bitcoin-exchanges-wont-tell-you-about-fees/`, [Online; Accessed: 2016-02-10]
6. Chaum, D.: Blind signatures for untraceable payments. In: Advances in cryptology. pp. 199–203. Springer (1983)
7. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Proceedings on Advances in cryptology. pp. 319–327. Springer-Verlag New York, Inc. (1990)
8. Constine, J.: After the Regretsy and Diaspora account freezes, we've lost confidence in PayPal. `http://techcrunch.com/2011/12/06/paypal-account-freeze/` (Dec 2011)
9. Dold, F., Totakura, S.H., Müller, B., Burdges, J., Grothoff, C.: Taler: Taxable anonymous libre electronic reserves
10. Dominguez, K.M.: Does central bank intervention increase the volatility of foreign exchange rates? Working Paper 4532, National Bureau of Economic Research (November 1993), `\url{http://www.nber.org/papers/w4532}`
11. Dunn, J.E.: Eurograbber SMS trojan steals 36 million from online banks. `http://www.techworld.com/news/security/eurograbber-sms-trojan-steals-36-million\-from-online-banks-3415014/` (Dec 2012)
12. Ehrenberg, B.: How much is your personal data worth? `http://www.theguardian.com/news/datablog/2014/apr/22/how-much-is-personal-data-worth` (April 2014)
13. European Central Bank: Third report on card fraud. `https://www.ecb.europa.eu/pub/pdf/other/cardfraudreport201402en.pdf` (February 2014)
14. European Central Bank: Our money. `http://www.new-euro-banknotes.eu/` (2016)
15. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. CoRR abs/1311.0243 (2013), `http://arxiv.org/abs/1311.0243`
16. Green, M., Miers, I.: Bolt: Anonymous payment channels for decentralized currencies. Cryptology ePrint Archive, Report 2016/701 (2016), `http://eprint.iacr.org/2016/701`

17. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on Bitcoin's peer-to-peer network. In: Proceedings of the 24th USENIX Conference on Security Symposium. pp. 129–144. SEC'15, USENIX Association, Berkeley, CA, USA (2015), http://dl.acm.org/citation.cfm?id=2831143.2831152
18. Holz, R.: Empirical analysis of Public Key Infrastructures and investigation of improvements. Ph.D. thesis, TU Munich (2014)
19. Jeffries, A.: Why don't economists like Bitcoin? (2013), http://www.theverge.com/2013/12/31/5260534/krugman-bitcoin-evil-economists, [Online; Accessed: 2016-02-28]
20. Jones, R.: Cap on card fees could lead to lower prices for consumers. http://www.theguardian.com/money/2015/jul/27/cap-on-card-fees-retailers (July 2015)
21. Kahn, C.: May 2014 financial security index charts (2014), http://www.bankrate.com/finance/consumer-index/financial-security-charts-0514.aspx, [Online; Accessed: 2016-02-10]
22. van Kersteren, A.: Cross-origin resource sharing. http://www.w3.org/TR/cors/ (January 2014)
23. Kügler, D.: On the anonymity of banknotes. In: Privacy Enhancing Technologies. pp. 108–120. Springer Verlag (2004)
24. Lehmann, C.: Bitcoin: Digital fool's gold? (2015), http://inthesetimes.com/article/17859/bitcoin-the-rush-for-digital-fools-gold, [Online; Accessed: 2016-02-28]
25. Lewis, N.: Bitcoin is a junk currency, but it lays the foundation for better money (2013), http://www.forbes.com/sites/nathanlewis/2013/05/09/bitcoin-is-a-junk-currency-but-it-lays-the-foundation-for-better-money/, [Online; Accessed: 2016-02-28]
26. Malmo, C.: Bitcoin is unsustainable (2015), https://www.cryptocoinsnews.com/what-bitcoin-exchanges-wont-tell-you-about-fees/, [Online; Accessed: 2016-02-10]
27. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: Anonymous distributed e-cash from Bitcoin. In: Security and Privacy (SP), 2013 IEEE Symposium on. pp. 397–411. IEEE (2013)
28. Murdoch, S.J., Anderson, R.: Verified by visa and mastercard securecode: Or, how not to design authentication. In: Proceedings of the 14th International Conference on Financial Cryptography and Data Security. pp. 336–342. FC'10, Springer-Verlag, Berlin, Heidelberg (2010), https://www.cl.cam.ac.uk/~rja14/Papers/fc10vbvsecurecode.pdf
29. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
30. NYA International: Cyber extortion risk report 2015. http://www.nyainternational.com/sites/default/files/nya-publications/151027_Cyber_Extortion_Risk_Report_2015_0.pdf (October 2015)
31. Perlman, M.: The Invention of Capitalism: Classical Political Economy and the Secret History of Primitive Accumulation. Duke University Press Books (2000)
32. Reid, F., Harrigan, M.: An Analysis of Anonymity in the Bitcoin System. In: Altshuler, Y., Elovici, Y., Cremers, A.B., Aharony, N., Pentland, A. (eds.) Security and Privacy in Social Networks, pp. 197–223. Springer New York (2013), http://arxiv.org/abs/1107.4524
33. ibi research: Digitalisierung der gesellschaft 2014 — aktuelle einschätzungen und trends. http://www.ecommerce-leitfaden.de/digitalisierung-der-gesellschaft-2014.html (2014)
34. Riley, M., Elgin, B., Lawrence, D., Matlack, C.: Missed alarms and 40 million stolen credit card numbers: How Target blew it. http://www.bloomberg.com/bw/articles/2014-03-13/target-missed-alarms-in-epic-hack-of-credit-card-data (March 2013)
35. Rundle, G.: The humble credit card is now a political tool. http://www.crikey.com.au/2011/10/25/rundle-humble-credit-card-now-a-political-tool-just-ask-wikileaks/ (Oct 2011)
36. Stallman, R.: How much surveillance can democracy withstand? WIRED (2013)
37. Sweney, M.: City AM becomes first UK newspaper to ban ad blocker users. http://www.theguardian.com/media/2015/oct/20/city-am-ban-ad-blocker-users (October 2015)
38. Szent-Ivanyi, T.: Wie firmen ihre kassen manipulieren. http://www.fr-online.de/wirtschaft/steuerhinterziehung-wie-firmen-ihre-kassen-manipulieren-,1472780,31535960.html (August 2015)
39. Trautman, L.J.: Virtual currencies; Bitcoin & what now after Liberty Reserve, Silk Road, and Mt. Gox? Richmond Journal of Law and Technology 20(4) (2014)
40. Volckart, O.: Early beginnings of the quantity theory of money and their context in polish and prussian monetary policies, c. 1520-1550. The Economic History Review 50(3), 430–449 (1997), http://www.jstor.org/stable/2599810
41. W3c: Web payments payment flows. https://github.com/w3c/webpayments/tree/gh-pages/PaymentFlows (February 2016)
42. Wright, S.: PCI DSS A Practical Guide to Implementing and Maintaining Compliance. It Governance Ltd, 3rd edn. (2011)