

# Portal: Micropayments on the Paywalled Internet

Samvit Jain

Advisor: Brian Kernighan

Department of Computer Science, Princeton University

## **Acknowledgements**

I would like to thank Professor Brian Kernighan for his thoughtfulness and dedication as my advisor in this year-long effort, Professor Arvind Narayanan for first working with me on the idea of micropayments as my junior independent work advisor, and my parents and sister for their love and support through all these years.

## Abstract

*In this work, we propose and evaluate Portal, a payment protocol and software system that enables one-click purchases of long-form news content on the Internet, without requiring a user to sign up for a subscription or login to a content provider’s website. The payment protocol enables clients to purchase a digital good (e.g. a single news article) via a standard payment mechanism, such as PayPal or a credit card, and then claim the good from the content provider over an unauthenticated HTTP channel by providing a valid proof-of-payment. This proof demonstrates that 1) a payment transaction of sufficient value was issued for the particular good being claimed (article id verification), and 2) the identity of the payer matches the identity of the client claiming the good (user id verification). Our proposed client-side software handles the construction and provision of this proof, taking the place of manual, password-based authentication.*

*Our research is motivated by the failure of most news sites to convert a significant percentage of their online readership base to paying subscribers, and various identified shortcomings in the paid subscription model. We present the technical architecture for an alternate monetization system, which allows the purchase of web content on a case-by-case basis, does not lock readers in to a long-standing contract, and does not stipulate that users log in to purchase and read an article discovered while browsing the web. After detailing the payment protocol underling our proposed system, we discuss our particular software implementation, with a focus on key design choices, such as the use of PayPal as a central transaction data store, verified email addresses as universal user identifiers, and client-side certificates for user authentication.*

*We conclude by evaluating our system on the basis of usability concerns, privacy and security considerations, and questions of adoption and scalability. We then outline key facets of a deployment and launch plan for Portal, ending with a discussion on potential avenues for future work.*

# 1. Introduction

One of the key facets of journalism's struggle to adapt to the digital age has been the battle to replace declining print revenues with income from digital content. News sites have traditionally monetized online content in two ways - display advertising and paid online subscriptions. The increasing concentration of digital advertising spending in five technology companies,<sup>1</sup> the low per-user revenue figures characteristic of display advertising, and the rise of ad blockers on desktop and mobile devices, however, have convinced many news companies that only one of the two monetization models will be ultimately sustainable [28, 32]. Case in point: in March 2011, The New York Times took the bold step of erecting a subscription paywall, limiting its readers access to 20 free articles per month [30].<sup>2</sup> In doing so, the Times made the key bet that its future business would ride on the contributions of loyal readers, not advertisers.

Attracting paid subscribers, however, has proved incredibly difficult for all but the largest players in the online news space. While The New York Times can boast of 1.4 million digital-only subscribers, of 57 million monthly-active readers [21, 23], Tronc (formerly, Tribune Publishing) - the owner of The Los Angeles Times, The Chicago Tribune, and The Baltimore Sun - counts only 115,000 digital-only subscribers across its entire portfolio [24]. Despite its sizable online subscription base, however, even The New York Times has struggled to reach profitability. Paid subscribers form 2.4% of the sites's monthly active readers, but contribute 56% of its digital revenue, a significant figure but not enough to reliably keep the Times out of the red: in the first quarter of 2016, the New York Times reported losses of \$14 million [21].

We identify three key challenges with the subscription model - inflexible pricing, cognitive overhead for users, and high upfront fixed costs that deter adoption, problems we discuss in greater detail in Section 2. Their consequences are two-fold: 1) users who may be willing to pay for a subscription do not do so, because the value of the subscription is difficult to ascertain, and is perceived to be lower than the financial and time costs of signing up for one, and 2) users who

---

<sup>1</sup>Google, Facebook, Yahoo, Microsoft, and Twitter.

<sup>2</sup>Now, 10 free articles per month.

may be willing to pay for specific journalistic pieces, but not enough in total to merit purchasing a subscription, do not have any other option but to settle for free content.

As mentioned previously, the subscription model is only one of several strategies available to online news sites to monetize digital content. In fact, news sites can be grouped by monetization model into at least four distinct categories:

1. **Free, ad-supported only** (e.g. CNN, Forbes, Bloomberg, Huffington Post)
2. **Free, ad-supported with subscription/paid membership option** (e.g. Wired, Guardian, Slate)

Some site content is free and ad-supported, but the remainder is only available to subscribing users; alternatively, users may be offered a paid membership option, which entails access to a premium, ad-free version of the site.

3. **Metered model** (e.g. The New York Times, Chicago Tribune)

Offers a fixed number of free articles, with a subscription required afterwards.

4. **Hard paywall** (e.g. The Wall Street Journal, Financial Times)

Offers no free articles; a subscription is required to access any content.

Portal seeks to introduce a new monetization model for news sites in categories (2), (3) and (4), and in particular those that have struggled to attract a critical mass of paid subscribers. Key components of this model, and the supporting infrastructure, include:

1. **Pay-per-article pricing** - Instead of paying for access to all of a news site's content for one month or one year, as with a subscription, users will have the option to purchase *individual articles*. This will involve a decision on the part of news companies to unbundle their content, and require sites to set prices for individual stories.
2. **Site-agnostic payments** - To purchase an article from a news site, users need not create an account, or enter into any kind of prior contract with the site. Users will be able to make one-time payments to any news site supporting our proposed payment mechanism, via a single piece of identifying information: a verified email address.
3. **One-click payment flows** - Having authorized future small-value payments through a central

account, users will be able to purchase articles from news sites via a single click in their web browser, without logging in to the provider site. Moreover, the article will be available for reading within seconds of purchase.

Through these features, Portal addresses each identified problem in the subscription model: 1) pay-per-article schemes allow for much finer-grained pricing of content, enabling the market entry of users who would not normally pay for news, 2) site-agnostic payments reduce the upfront time cost of becoming a subscriber, and the cognitive overhead of managing accounts with multiple sites, and 3) one-click payment flows eliminate the need to log in to a news site to read an article, and reduce the process of purchasing a news article to a single extra step (beyond what it would be if an article was pre-paid for via a subscription).

Of note, many sites in category (1), which currently rely on banner and video advertising for monetization, are also struggling financially, in part due to the rise of ad blockers,<sup>3</sup> and could also be candidates for a new revenue model based on direct payments from users. But as refund rates on content sold by Blendle, a news portal service experimenting with pay-per-article pricing (see Section 3.1), demonstrate, while users may be willing to pay for long-form journalism and original reporting, other kinds of content, including basic daily news, tabloid, and "listicles", may be hard to sell to users at any price [18]. The content of sites in category (1), being generally shorter, more time-sensitive, and optimized for broad readership, is likely thus a weaker fit for a user payments-based monetization scheme. Noting, however, that the connection is a two-sided one, with monetization considerations influencing the type of content a site hosts, it may certainly be the case that the availability of an alternate business model would effect an evolution in the content of category (1) sites, and a subsequent adoption of the new monetization scheme.

---

<sup>3</sup>70 million people in the U.S. use ad blockers as of 2016, or 26% of Internet users. This is almost twice the number that did in 2014 [4]. According to one estimate, ad blocking cost publishers nearly \$22 billion in 2015 [32].

## 2. Motivation

### 2.1. The Subscription Model

Becoming a subscriber to an online news site entails several challenges for users. Firstly, most news sites price content at two discrete points - at the extreme low end, where content is free, ad-supported, and tailored for mass readership, and at the high end, where sites offer premium content and aim to maximize revenue on the market segment willing to pay for online news. The New York Times' subscription offerings, for example, are priced starting at \$195 per year for a basic digital access package, while subscriptions for the Financial Times range from \$335 to \$610 per year [13, 9]. This highly discretized form of price discrimination results in a loss of business from users who may be willing to pay for select content, or for some number of articles per month, but not enough to merit purchasing a subscription. Moreover, many users do not exclusively rely on a single source for news, and may be unwilling to commit to paying a single site for news content by purchasing a subscription.

Secondly, signing up for and managing subscriptions involves *cognitive costs* for users. Users must determine which set of sites are worth subscribing to, create accounts with and store credit information with those sites, and then manage renewals and auto-renewals over the lifetime of their subscription membership. While users are used to managing subscriptions for music (Spotify) and television (Netflix), both these examples represent centralized streaming services that offer access to a large and growing fraction of all available content in their respective genres. In contrast, paid, online journalism is characterized by a relative lack of centralization - while The Wall Street Journal broadly covers politics and business, The Economist specializes in global news; while The New Yorker focuses on culture and social issues, Vogue centers on fashion and style. This poses a conundrum for users when it comes to subscriptions, forcing a choice between convenience and content variety.

Finally, access to subscription content is generally restricted via a *paywall*, a web interface that stipulates that users log in to, or create, a paid account to read a particular article. Non-subscribers

interested in trying out a subscription, either by actually purchasing one or by participating in a free or discounted trial offering, must go through the process of setting up a paid account, which involves providing, at minimum, basic personal details, credit card information, and a billing address. (See, for example, the subscription signup form for the *Financial Times* - Fig. 1.) This process itself deters users from experiencing the value of a paid subscription: a user interested in reading a single article is asked to pay the high, fixed time cost of acquiring a year's worth of content. Since news content is an *experience good*, whose value is hard for users to ascertain before consumption, users may underestimate their willingness to pay for a subscription, especially when faced with the relatively clearer costs of becoming a subscriber [22].

## **2.2. News site revenue**

In the first quarter of fiscal year 2016, The New York Times earned \$42 million in digital advertising revenue from 57 million monthly-active readers, and \$54 million in digital subscription revenue from 1.4 million subscribers [21]. Paid subscribers thus form only 2.4% of the site's active online readers, but contribute 56% of The New York Times' digital revenue. These proportions are a reflection of the structure of the "freemium" business model – a large number of readers are offered some access to the Times' content for free in order to generate interest in the subscription offering; a small percentage actually convert to paying subscribers and contribute the bulk of the site's revenue.

The fact that access to news content is a divisible service, with the smallest feasible unit being an individual article, suggests the viability of alternate pricing models. If The New York Times were to lease individual articles to readers, instead of selling access to its content in a renewing subscription format, could it earn more revenue from its monthly-active readers, without cannibalizing its subscription base? As a conservative, back-of-the-envelope calculation, if the Times' 57 million non-subscribers purchased 2 articles per month on average, priced at \$0.50 each, it would be replacing \$42 million in quarterly advertising revenue with \$170 million in direct payments from users, a more-than-four-fold increase in revenue. Beyond offering the ability to better serve active, non-subscribing readers, per-article pricing also opens up the possibility of fine-grained price



# FINANCIAL TIMES

YOU'VE SELECTED **STANDARD FT.COM**

Email address\*

This address will be used to create your account

Enter your email address

Password\*

6 characters min (numbers & letters only)

Enter a password

☐ Show password

First Name\*

Last Name\*

First name

Last name

Phone number\*

5 to 15 characters (numbers only)

Enter your phone number

CUSTOMISE YOUR EXPERIENCE

In which industry do you work?\*

Please select an industry

Which of the following best describes your job responsibility?\*

Please select a job responsibility

What's your job position?\*

Please select a job position

PAYMENT DETAILS

Standard FT.com

☒ **Annual option**  
**\$6.45 per week**  
Single \$335.40 payment\*

☐ **Monthly option**  
**\$36.00**

Billing Country\*

United States

Billing State\*




Please select a state

Billing Zip Code\*

Zip code


Secure payment

Credit Card

☐ ☒ ☐ 


Cardholder Name\*

As it appears on the card



Card Number\*

The long number across your card



Expiration Date (MM/YYYY)\*

- Select One -

/

- Select One -

Security code\*

TERMS AND CONDITIONS

☐ \* I confirm I have read and agree to the [terms & conditions](#), [privacy policy](#) and [cookie policy](#).  
\* Your subscription lasts for 1 year. To give you uninterrupted access we will automatically renew it using the payment method you provided, unless you cancel before your renewal date. We will send you a reminder of this date in advance.  
Find out more about our cancellation policy in our [terms & conditions](#).

We may contact you occasionally, by email and phone, to improve your experience of the Financial Times.

☒ By submitting this form, you indicate your consent to also being contacted by Financial Times by email, post or phone about our other products, services or special offers unless you untick this box.  
You can change your preferences anytime in the My Account section.

Buy now

Figure 1: Subscription Signup Form for the *Financial Times* [10]

9

discrimination, where users are charged based on their willingness to pay (as estimated by their location, income, and past behavior, among other indicators). Though such a pricing model could be controversial, it would allow the Times to maximize the revenue it earned from its readers, unlike a fixed price point approach, such as the paid subscription model, which invariably results in a loss of business from some subset of potential customers.

Selling content by the article is not the only monetization model available to news sites. Other alternatives include selling day-long or week-long passes to the digital version of a site, charging only for certain kinds of content (e.g. exclusive coverage of a current event, live video, opinion pieces), or even using per-article pricing as a funnel for subscription conversions. While we choose to focus on per-article pricing in this paper, our discussion, and the technical infrastructure we propose, is applicable to these other pricing models as well. Our goal is not argue for a particular monetization scheme, but to illustrate the general technology and business considerations that underlie the leasing of content for small-value, non-contractual payments from users.

### **3. Related Work**

#### **3.1. Services**

**Blendle** is a news portal service that allows users to purchase journalistic content from partner news sites on a per-article basis. Blendle was launched in the Netherlands and Germany in 2015, where it amassed 650,000 users, and in the U.S. in 2016, and cites backing by prominent publishers, including The New York Times Co. and German media giant Axel Springer, which have co-invested \$3.7 million in the company [18]. The service cites 1 million registered users globally as of August 2016, of which 150,000 are paying customers [25]. Blendle charges between \$0.09 and \$0.59 for individual articles, and offers users the option to refund their payment if an article does not meet their expectations [25]. Despite this, it claims a surprisingly low overall refund rate of 10% [18].

Two key takeaways from Blendle’s experience are 1) that unbundled pricing schemes for online journalism may in fact be viable, and 2) that users are much more willing to pay for certain kinds of content (“original reporting, great background stories, deep analysis, big profiles, and interviews”),

than others (news easily available elsewhere for free, tabloid articles, listicles) [18]. This is seen in the significantly higher refund rates, as measured by Blendle, on the second content category [18].

**PayPal** is a company and associated online payments system that supports money transfers between any two entities on the web with verified email addresses. This includes both money transfers between peers, and payments from a buyer (PayPal personal account) to a merchant (PayPal merchant account) [16]. PayPal serves as the backing payments system for the prototype implementation of Portal described in this work, as it enables site visitors (buyers) to make payments to site hosts (merchants), knowing simply an email address associated with the site's merchant account. Furthermore, PayPal offers users the ability to *preapprove* a number of payments of a specified individual and collective maximum value (e.g. \$0.50 and \$100) over a certain time period (e.g. 1 year) [5]. This PayPal feature allow third-party applications to automatically charge a user's PayPal account whenever a user indicates interest in purchasing a digital good, without redirecting to PayPal. Email address-based payments and payment preapprovals together eliminate the need for a user to 1) store credit card information with every news site they wish to purchase content from, or 2) explicitly authorize a payment of small value (e.g. \$0.30) via a multi-step process.

PayPal is not a *replacement* for Portal, because while it offers the necessary email-to-email money transfer functionality, it does not arbitrate the transfer of the associated good. With PayPal, a user typical prepay for a good, which the merchant then emails, ships, or otherwise transfers to the user out-of-band, after confirming the receipt of payment. If the "good" is an HTTP endpoint, accessed by a user via a web browser, then the merchant (news site) must associate the HTTP request for the endpoint with a received PayPal payment, in order to ensure that the correct user is gaining access to the web resource. While a PayPal payment is identified via the sender's email address, an HTTP request contains by default only proxies for user identity (e.g. user agent string, browser plugins, system fonts, etc.) In this paper, we show how a user can definitively prove ownership in their HTTP request of the email address associated with a payment, without logging into the site. This proof is encapsulated in a novel security protocol, which forms a key feature of the Portal service.

## 4. Payment Protocol

### 4.1. HTTP 402 Protocol

Our payment verification protocol builds on a basic HTTP protocol for requesting and receiving digital goods from a server, called the HTTP 402 protocol (see Fig. 2). The protocol consists of a dual request-response cycle [26]:

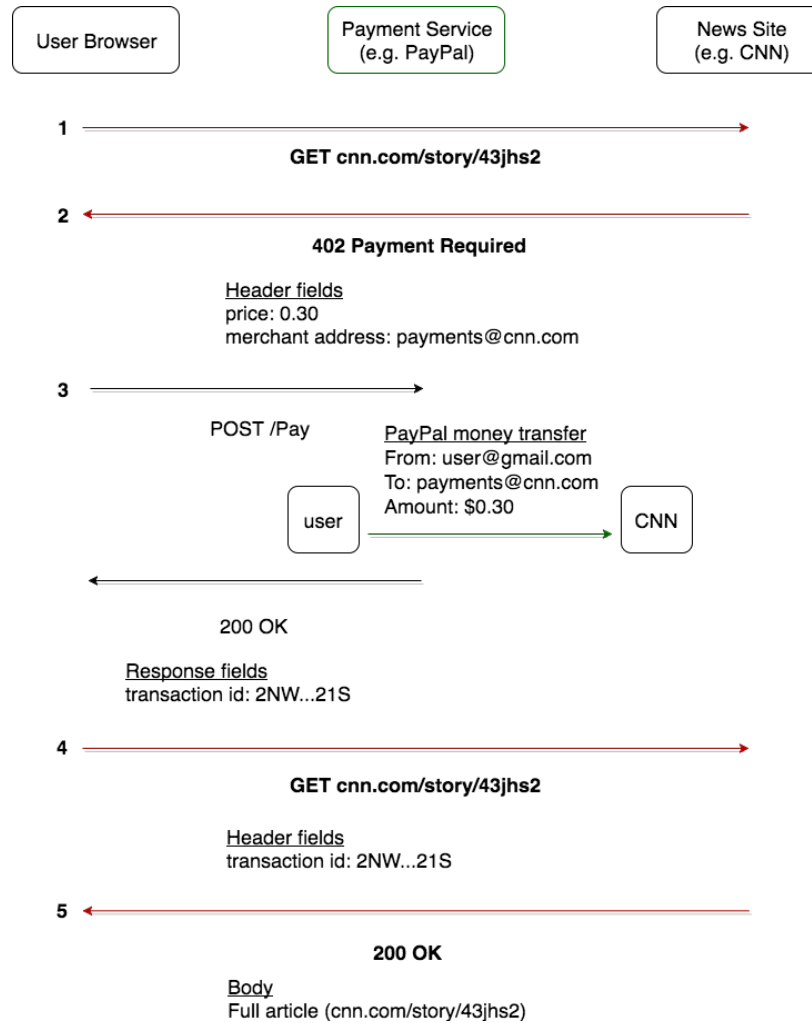
1. The user makes an HTTP GET request for a web endpoint (e.g. `cnn.com/story/43jhs2`)
2. The server returns an HTTP 402 Payment Required response, signaling that payment is required to access the endpoint. The body of the response contains a preview of the article, while the header contains *instructional fields*, which include the price of the endpoint and the merchant account address to which to direct payment.
3. The user makes a payment to the merchant over a different channel (e.g. PayPal, Stripe, credit card network, Bitcoin) via an API call to the corresponding service. The service returns a *transaction id X*, a string that uniquely identifies the purchase event.
4. The user retries the HTTP GET request for the endpoint. This second request contains the transaction id *X*, along with a "proof-of-payment" field in the HTTP header that attests responsibility for the purchase transaction *X* just made.
5. The server returns a 200 OK response, along with the requested resource (e.g. the full article), after validating the user's proof-of-payment.

### 4.2. Payment Verification - Context

How does a news site verify that a transaction id corresponds to a payment made by the user supplying it, for the article being requested? In particular, a user could publish a transaction id, allowing others to use it, or reuse the transaction id for different articles. Preventing these exploits requires amending the HTTP 402 Protocol with a *payment verification* step, by which a news site verifies the "proof-of-payment" supplied by users.

At a higher level, we would like to enable a content provider (e.g. a news site) to verify that a

**Figure 2: Payment Protocol - Basic Setup**



user making an HTTP request for a web endpoint (e.g. a news article) paid for it at some point, without 1) requiring the user to explicitly authenticate via a username and password, or 2) storing any payment-related state at the provider site. We are willing to trust "reputable" third-parties, such as PayPal, to proxy as data stores for the news site. We are also willing to rely on non-reputable third-parties to store data (e.g. a newly launched web service, such as our own), granted that we can verify the integrity of data returned by the service.<sup>4</sup>

The payment verification problem has two components. When a user requests an article, and supplies a transaction id to claim it, the news site must confirm that:

1. the transaction id corresponds to the article being claimed (**article id verification**)

<sup>4</sup>Note that data integrity has two parts to it in this context: 1) verifying that a record was not tampered with (or deleted), and 2) verifying that a record was actually submitted by some legitimate participant in the system.

2. the transaction id corresponds to a purchase this user made (**user id verification**)

To better illustrate this problem, consider the universe of all purchases from a particular news site as a large 2-dimensional matrix, whose rows represents all users in the ecosystem, and whose columns represent all articles sold by the site. (see Table 1)

**Table 1: Universe of Transaction Ids**

	1	2	3
(A)lice		tx-3232	tx-2812
(B)ob	tx-5283	tx-8404	
(C)arol		tx-1287	

If a particular user  $U$  ever paid for a particular article  $A$ , a transaction id should appear in cell  $(U, A)$ . A blank cell, in contrast, represents an article that a particular user never purchased.

We begin by assuming the existence of a secure, trusted key-value store that records all transactions at the time of purchase, and offers an interface for the news site to look up a transaction id, and retrieve the associated user and article ids. The key-value store thus holds mappings from transaction id to  $(U, A)$  tuples.

Then, when a user  $U'$  supplies a transaction id  $X$  to claim an article  $A'$ , the news site's verification obligation is as follows: it must look up the transaction id  $X$ , yielding a tuple  $(U, A)$ , and then verify that  $A'$  in fact equals  $A$  and  $U'$  in fact equals  $U$ .

The key challenge is that the news site cannot rely on a user to faithfully self-report its user id  $U'$ . If Bob wanted to claim an article paid for by Alice, and Alice shared the transaction id of the purchase with him, Bob could simply supply  $(U_{Alice}, A)$  instead of  $(U_{Bob}, A)$  with his request. Clear incentives exist for users to collude - Alice and Bob could each purchase a number of articles separately, and then exchange the transaction ids for their purchases, allowing both to read more articles than they individually paid for. A mechanism must thus be developed by which a user can *prove* to the news site its user id  $U'$ . This is the **user id verification** problem.

A user supplies an article id  $A'$  implicitly by making an HTTP request for article  $A'$ . This

means that the user cannot "lie" about  $A'$  (the article id is not self-reported), reducing the **article id verification** problem to that of ensuring that the key-value store indeed guarantees data integrity. In particular, if mappings in the key-value store cannot be tampered with, then a given transaction id  $X$  can only be associated with a single article id  $A$  over its entire lifetime, a mapping fixed at the time of purchase. Then the only article the user can request using  $X$  is article  $A' = A$ .

### 4.3. Payment Verification - User id

Our solution to the user id verification problem involves the provision of a two-part proof (along with a transaction id) by the user to the news site on its second HTTP request for an article:

1. A user **public key certificate** (PKC), obtained from a reputable Certificate Authority (CA), that attests that a particular public key,  $PK_{Alice}$ , belongs to a user with a given user id,  $U_{Alice}$ . More precisely, the PKC consists of the tuple  $(U_{Alice}, PK_{Alice}, sig_{CA}(U_{Alice}, PK_{Alice}))$ . A valid signature by the CA (verified via the CA's public key) ensures that the certificate is untampered, and thus that the user id-public key mapping is legitimate.
2. A **digital signature** by the user on a unique **request id** assigned by the news site, that attests that the requesting user owns the private key  $SK_{Alice}$  corresponding to the public key  $PK_{Alice}$  contained in the certificate.<sup>5</sup> This request id is assigned by the news site on the user's first HTTP request; a valid signature is expected on the user's second request.

The news site's final verification obligation is to lookup the supplied transaction id  $X$  in the trusted key-value store, and verify that the associated user id,  $U_X$  (a mapping stored at the time of purchase), is identical to the user id in the certificate,  $U_{Alice}$ . This yields the following chain of association:

$$SK_{Alice} \rightarrow PK_{Alice} \rightarrow U_{Alice} = U_X \rightarrow X$$

The news site performs three checks, which establish the following:

1. The user owns a private key  $SK_{Alice}$  corresponding to public key  $PK_{Alice}$  (user signature check)

---

<sup>5</sup>Alternatively, the user could have obtained the signature from the actual owner of the private key, by providing his/her assigned request id. See Section 4.6 for a discussion of this exploit.

2. The public key  $PK_{Alice}$  corresponds to user id  $U_{Alice}$  (certificate check)
3. The user id  $U_{Alice}$  in the certificate corresponds to the supplied transaction id  $X$  (KV-store lookup)

Together, these checks assert that a user requesting an article is indeed the same user as the one who made the purchase (as identified by the transaction id).

This procedure successfully protects against the following category of attacks:

#### **Publish-Replay Attack**

1. Alice requests article  $A$  from a news site, and is assigned request id  $r$ .
2. Alice purchase article  $A$ , and is assigned transaction id  $X$  for her purchase.
3. Alice publishes her public certificate  $PKC_{Alice}$  and her signature  $sig_{Alice}(r)$  on a public forum.
4. Bob, who hasn't purchased article  $A$ , discovers Alice's post, and requests  $A$  from the news site, providing  $X$ ,  $PKC_{Alice}$ , and  $sig_{Alice}(r)$  in his request.

This will fail because Bob will be assigned a different request id  $r'$  by the news site, so the signature  $sig_{Alice}(r)$  will not check out. Indeed, if Alice were to re-request article  $A$  the following day, she herself would be assigned a different request id  $r''$ . Request ids are specific to client connections, requiring users to produce new signatures for different HTTP requests.

If Bob instead provides  $(X, PKC_{Alice}, sig_{Bob}(r'))$ , he will also fail the user signature check - his signature will not check out with the public key contained in the certificate, which is Alice's.

If Bob modifies the certificate  $PKC_{Alice}$  to contain *his* public key but Alice's user id,  $(X, PKC = (U_{Alice}, PK_{Bob}), sig_{Bob}(r'))$ , he will fail the certificate check - the CA's signature will not check out, and the news site will know the certificate has been tampered with.

Finally, if Bob provides  $(X, PKC_{Bob}, sig_{Bob}(r'))$ , perhaps the most straightforward exploit he could attempt, he will fail the KV-store lookup check - the user id associated with  $X$  will be  $U_{Alice}$ , not  $U_{Bob}$ , as contained in  $PKC_{Bob}$ , and the news site will reject the transaction id  $X$ .

#### **4.4. Payment Verification - Article id**

Article id verification serves to protect against the following exploit: a user makes a purchase for article  $A_1$ , which costs \$0.30, and uses the obtained transaction id  $X$  to claim articles  $A_2$  and  $A_3$ ,



which also happen to cost \$0.30. In the absence of any checks beyond a price comparison of the article being requested, and the value of the associated payment, a user could use a single \$0.30 purchase token (i.e. transaction id) to claim all \$0.30 articles offered by a news site. Solving this problem requires creating a fixed association between payment transactions and the good being purchased - namely, by storing the article id  $A$ , along with the user id  $U$ , for transactions in the trusted key-value store.

As stated before, if  $\text{tx-id-}X \rightarrow (U, A)$  mappings in the key-value store are immutable (i.e. the value field cannot be updated after the time of payment), then a transaction id  $X$  can only ever be associated with a single article id  $A$ . This in turn yields the desired property that a given transaction id can only be used by a user to claim a single article.

#### 4.5. Payment Verification - Full Protocol

We now present the full protocol for payment verification (see Fig. 3), which is comprised of the basic HTTP 402 Protocol, together with a payment verification procedure developed in this section:

1. The user makes an HTTP GET request for a web endpoint (e.g. `cnn.com/story/43jhs2`)
2. The server returns an HTTP 402 Payment Required response, signaling that payment is required to access the endpoint. The body of the response contains a preview of the article, while the header contains *instructional fields*, which include:
  - (a) the **price** of the article
  - (b) the **article id** for the requested article
  - (c) a **request id**  $r$  generated for the client connection
  - (d) a merchant account address to which to direct payment.
3. Case 1: If the user has not previously purchased the article, the user makes a payment for it via an API call to the appropriate payment service (e.g. PayPal, Stripe). The user's request contains the following fields:
  - (a) the article's **price** (money transfer amount)<sup>6</sup>

---

<sup>6</sup>Note that the user has no incentive to lie about the price of the article, because the user's account will be credited by that amount (and paying too little will cause the news site to reject the associated transaction id).

(b) the article's **article id** (money transfer memo)<sup>7</sup>

(c) the merchant account address of the news site (recipient address)

Case 2: If the user *has* paid for the article before, the user queries Portal for a record of the purchase, providing the domain (e.g. cnn.com) and article id (e.g. 43jhs2) of the article.

4. Case 1: If a new purchase is being made, the payment service executes the money transfer, and then stores the user's **user id**  $U$ , along with the article's article id  $A$  and price  $p$ , in the trusted key-value store. The key for this value tuple  $(U, A, p)$  is a newly generated **transaction id**  $X$  assigned by the payment service to this transaction.

Case 2: If the user is retrieving a record of an existing purchase, Portal queries for the transaction id associated with the provided article id (via a reverse lookup on the key-value store, or a forward lookup on a separate database it maintains).

In both cases, the user is returned an ACK (success or failure), along with the transaction id  $X$  of the purchase, if successful.

5. The user retries the HTTP GET request for the endpoint. This second request contains a full proof-of-payment in its header, composed of the following:

(a) the **transaction id** of the associated payment

(b) the user's **public key certificate** -  $(U_{user}, PK_{user}, sig_{CA}(U_{user}, PK_{user}))$

(c) the user's **digital signature** on the request id -  $sig_{user}(r)$

6. The server validates the user's proof-of-payment. This is composed of the following three user id checks, discussed earlier:

(a)  $sig_{CA}$  is valid for  $PK_{CA}$  (certificate check)

(b)  $sig_{user}$  is valid for  $PK_{user}$ , request id  $r$  (signature check)

(c)  $KV.get(X).U$  matches  $U_{user}$  in cert (KV store lookup check)

and the following two article id checks:

(a)  $KV.get(X).A$  matches article id  $A$  of endpoint

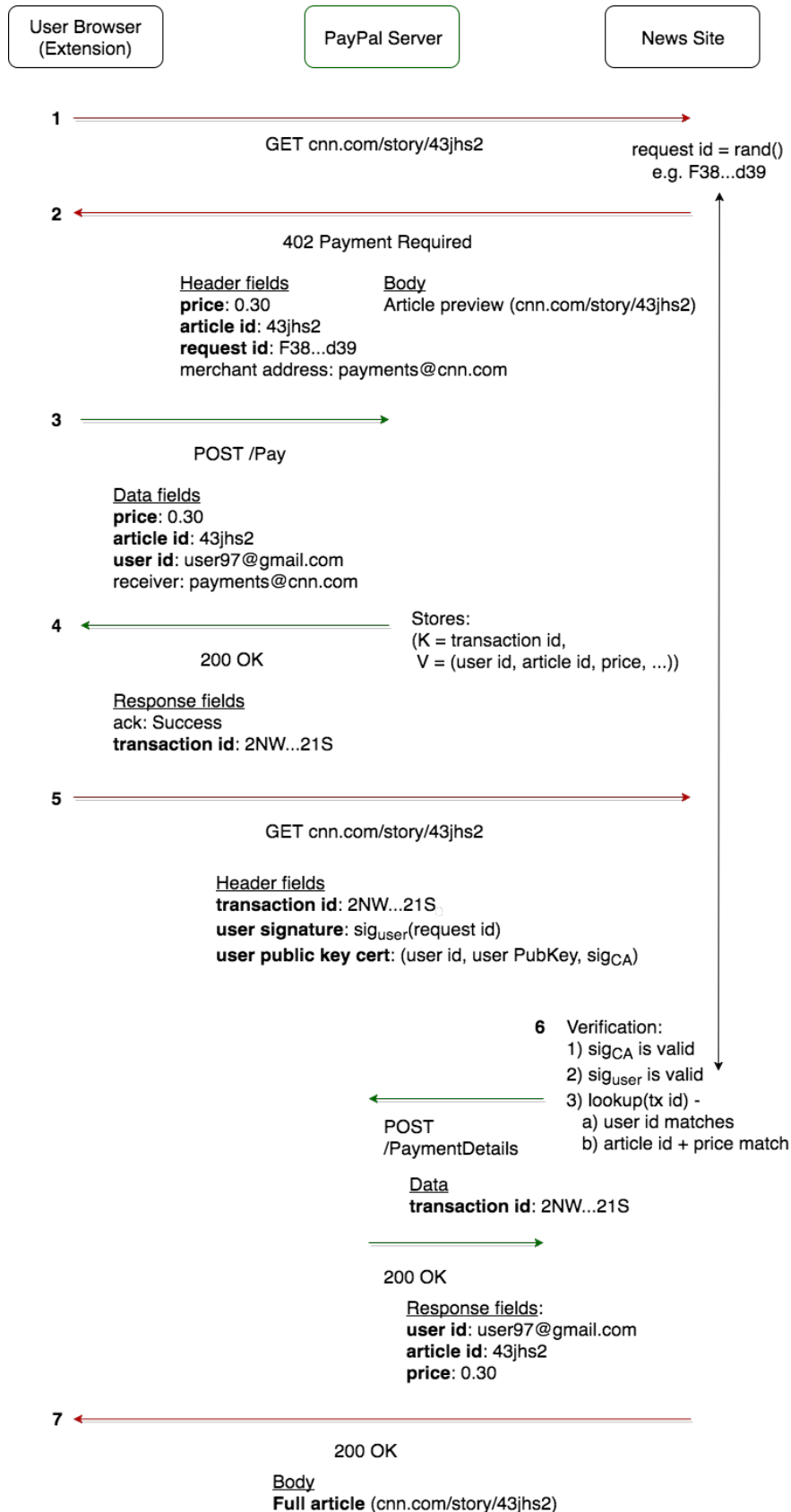
(b)  $KV.get(X).p$  matches price of article  $A$

---

<sup>7</sup>The user also has no incentive to misrepresent the article id; the news site will accept payment for any article it sells, as long as the amount paid matches the article's price.

7. If all of these checks pass, the news site returns a 200 OK response, along with the requested resource - the full purchased article.

**Figure 3: Full Payment Protocol**



## 4.6. Possible Exploits

While the verification procedure outlined protects against the Publish-Replay Attack discussed in Section 4.3, it does not prevent the following exploit:

### Signature Co-opt Attack

1. Alice purchases article  $A$  from a news site, and is assigned transaction id  $X$  for her purchase.
2. Alice sets up a web service in which she accepts as input a request id  $r$  and outputs her signature  $sig_{Alice}(r)$  on the request id. Along with this, Alice publishes her public certificate  $PKC_{Alice}$ .
3. Bob requests article  $A$  from the site, and is assigned request id  $r$ .
4. Bob submits  $r$  to Alice, and then provides  $X$ ,  $PKC_{Alice}$ , and  $sig_{Alice}(r)$  in his request to the news site, gaining access to an article Alice purchased.

The defining characteristic of this exploit is that Alice *issues signatures* for other users. Incentives exist for such an attack to be deployed; if articles generally cost at least \$0.30, Alice could charge a smaller amount, such as \$0.10, for generating a signature, and thereby implement an effective arbitrage scheme.

Note that this is a more sophisticated exploit than one in which Alice simply publishes some data, which is then used as-is by other actors to gain access to a resource. It does not scale as well as the Publish-Replay Attack, as it requires two-way communication between every colluding pair, and may thus not be as pressing a concern for news sites.

In fact, there exists an even simpler exploit that is not protected by the outlined verification protocol, and that is faced by any content provider wishing to lease access to a digital resource:

### Content Redistribution

1. Alice purchases article  $A$  from a news site, and is assigned transaction id  $X$  for her purchase.
2. Alice publishes the text of article  $A$  on a personal website.
3. Bob reads article  $A$  on Alice's website for free.

Such an attack would undoubtedly constitute copyright infringement, as evidenced by the excerpt of The New York Times' copyright noticed reproduced below:

## Copyright Notice

All materials contained on this site are protected by United States copyright law and may not be reproduced, distributed, transmitted, displayed, published or broadcast...

However, you may download material from The New York Times on the Web (one machine readable copy and one print copy per page) for your personal, noncommercial use only. [8]

Copyright law, however, has not prevented the proliferation of privacy involving books, music, and film online. News sites would thus have to invest effort into making articles difficult to reproduce, such as by splitting long-form content on multiple pages, and attempting to disable clipboard functions such as copy and paste on their webpages. These heuristics, along with the basic access control protocol outlined in this section, could together serve as a working solution to the problem of protecting content rights on the web.

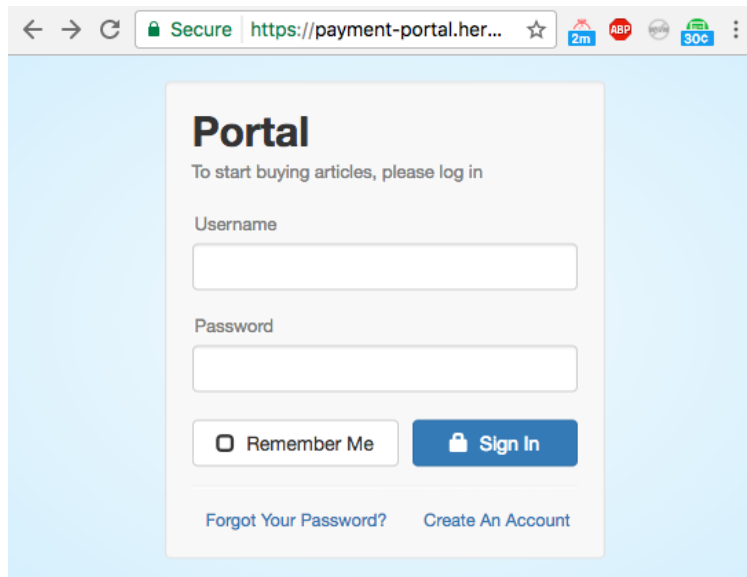
## 5. Implementation

### 5.1. Software Components

Portal consists of three key software components:

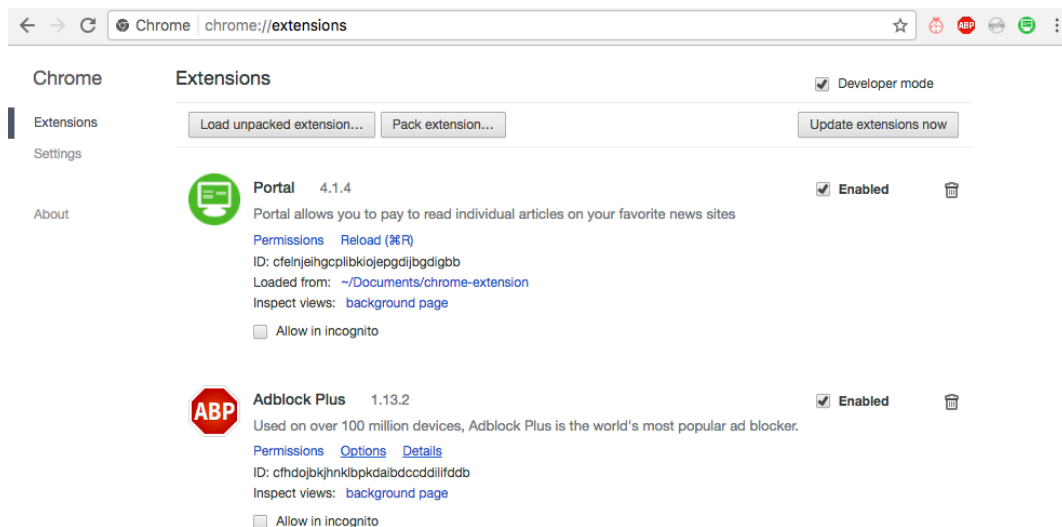
1. **Account service** - Implemented as a web application, the Portal account service stores payment information for users (i.e. PayPal payment authorization or credit card information), and records transactions to news sites. For users, the first step to purchasing news articles is to create an account with Portal, and link their PayPal account or add a credit card (see Fig 4). In our prototype, we implemented the account service as a Node.js web application running on Heroku. We currently only allow payments with PayPal, but anticipate adding support for credit card payments and Bitcoin in the future.
2. **Browser extension (client-side)** - The Portal browser extension enables users to make one-click payments for news articles discovered while browsing the web (see Fig 5). After creating an account with the Portal account service, users download the browser extension from the corresponding application store for their browser (e.g. Chrome Web Store for Google Chrome).

**Figure 4: Portal - Account Service**



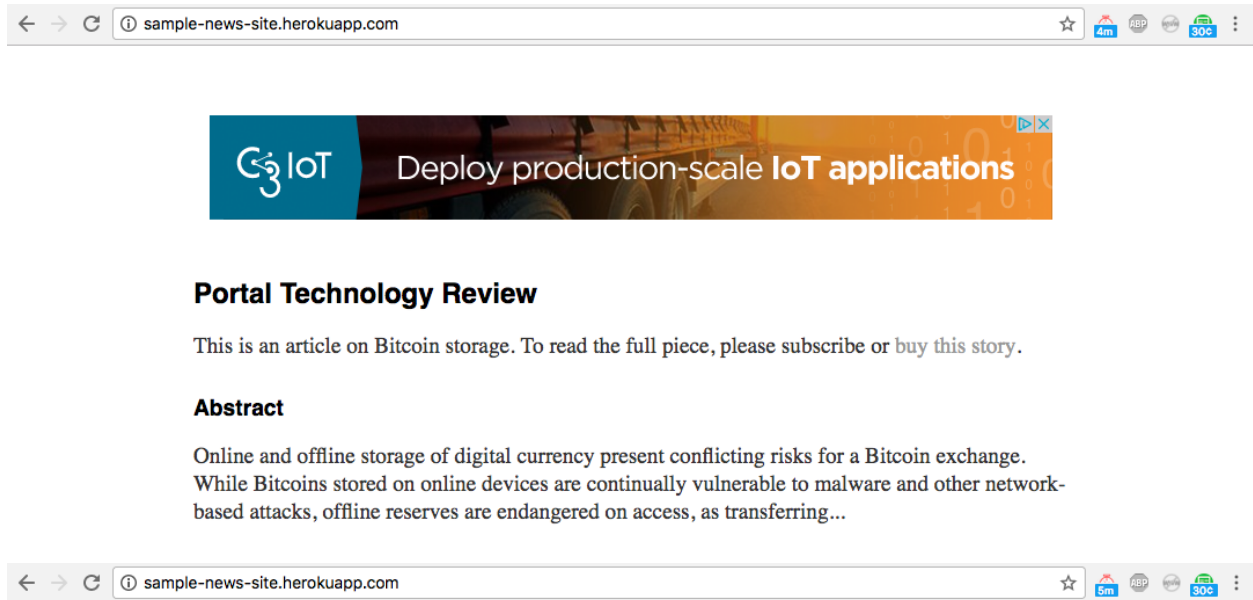
Day-to-day use of Portal consists mainly of interaction with the extension. Clicking on the extension while on an article preview hosted by a supporting news site results in a payment transaction to the site, and a reloading of the current page in the user's browser. If the transaction is validated, the user successfully gains access to the full article (see Fig 6). In our prototype, we implemented the browser plugin as a Chrome extension in Javascript.

**Figure 5: Portal - Chrome Extension**



3. **Payment verification code (server-side)** - Portal's payment verification code consists of logic run by news sites to verify that a valid payment was received from a user for an article requested

**Figure 6: Portal - Sample News Site**



by the user. In our prototype, payment verification logic was built in to our news site server implementation, as an HTTP request handler. For production implementations, the code could be packaged and distributed as a self-contained unit (e.g. a Node module for Node.js web applications) to be incorporated in a news site's server codebase.

## 5.2. Usage

### Setup

To use Portal, a user must first perform the following setup:



1. Create a PayPal account using a verified email address. (This will serve as the user's unique user identifier  $U$  in Portal's payment verification protocol.)
2. Create a Portal account, by providing an email address and creating a password, and then link the PayPal account created in the previous step. Linking a PayPal account involves logging into PayPal from within Portal, and then completing PayPal's *preapproved payments* authorization flow (see Figure 7). Doing so gives Portal the ability to spend up to a specified total amount (e.g. \$100) over a specified time period (e.g. 1 year), in payments up to a specified maximum value (e.g. \$1.00), from a user's PayPal account [5].
3. Install the Portal Chrome extension.

**Figure 7: PayPal Preapproved Payments Flow [5]**

The figure consists of two side-by-side screenshots of the PayPal Preapproved Payments Flow interface.

**Left Screenshot: Log in to your PayPal account**

- Language: English
- Header: Log in to your PayPal account
- Form fields: Email address, PayPal password
- Links: Problem with login?, Log In, Cancel
- Section: New to PayPal? Sign up and buy in a few easy steps.
- Footer: Consumer advisory – PayPal Pte. Ltd., the holder of PayPal's stored value facility, does not require the approval of the Monetary Authority of Singapore. Users are advised to read the terms and conditions carefully.

**Right Screenshot: Sign up for future payments**

- User: José Ramírez, Log out
- Header: Sign up for future payments
- Merchant: With: Pretty Flower Florist
- Start: July 7, 2012
- End: December 7, 2012
- Memo: Preapproval from test application
- Section: PayPal will use: Default payment method
- Text: By clicking the button below, I authorize Pretty Flower Florist to automatically charge my PayPal account for my future payments. If your payment fails, PayPal will use the other payment methods linked to your account. We will use your available PayPal balance first. View PayPal policies. To stop payments under this agreement, go to "My preapproved payments" in your PayPal Profile. To cancel this agreement, contact Pretty Flower Florist.
- Buttons: Agree and Continue, Cancel
- Footer: PayPal protects your privacy and security.

## Operation

Portal implements the HTTP 402 Protocol on the client and server sides, along with the verification protocol presented in this paper, to support non-authenticated payments for digital goods.

A user begins the process of making a payment to a news site by visiting the URL for an article in their browser. If the article is not free, the news site returns an HTTP 402 Payment Required

response, along with a preview of the article in the response body. The news site's response additionally contains the following fields: 1) the price of the article, and 2) a merchant address to which to direct payments for the article.

The user indicates interest in purchasing the article by clicking on the Portal browser extension, which triggers a POST request to a PayPal server endpoint. This results in a money transfer from the user's personal PayPal account to the news site's merchant account, identified by email addresses included in the user's POST request. If the transaction is successful, PayPal returns to the user a 200 OK response, along with a *transaction id*, a purchase token used by the user to identify the payment made for a particular article.

The Portal browser extension now reloads the current page, triggering a second HTTP GET request to the news site, which contains the retrieved transaction id along with a *proof-of-payment* (see Section 4). If the news site is able to successfully validate the proof-of-payment, it returns a 200 OK response, along with the full article initially requested by the user.

### 5.3. Previously Purchased Articles

On the client side, the Portal browser extension records every purchase a user makes in a private key-value store, data that is persisted on the Portal server. This allows Portal to look up the details of previously purchased articles. By querying the key-value store for an article URL, Portal can retrieve the transaction id identifying its purchase, if a user has paid for the article in the past. (If the article URL isn't found, this indicates that the user hasn't purchased the article yet, and a new payment transaction must be issued.) This transaction id, along with the appropriate proof-of-payment, is supplied to the news site in the second HTTP request for the article in the HTTP 402 Protocol.

### 5.4. Design choices

Our implementation of Portal involved making several key design choices:

1. **PayPal as a data store** - Section 4 emphasizes the dependence of the outlined verification protocol on a secure, trusted key-value store to hold transaction data, which can reliably serve insert requests from users and lookup requests from news sites. Users depend on this data store

to accurately record new transactions. If a user spends money to purchase an article, she would like to guarantee either access to the corresponding article or a refund of her payment, both of which depend on reliable inserts. News sites depend on this data store to return accurate information about transactions. A news site would like to guarantee that only records for actual payment transactions are returned (i.e. no made up or tampered records). Doing so requires the data store to function according to contract, and for the provider of the data store to act in a way that the news site expects. Ideally the provider is an entity with incentives compatible with those of the news site, and sufficient, independent reputation, so that collusion with users to achieve arbitrage is not a realistic possibility.<sup>8</sup>

Given these requirements, we chose to use the payment service itself (PayPal in our implementation) as an implicit key-value store. This solves the problem of reliable inserts in a natural way. When a payment service executes a money transfer, authorized by a user, it also stores a record of the transaction as a normal part of its operation. Users can count on this record to be consistent with the actual payment made, as this is a fundamental service guarantee offered by payment processors. In particular, a user can expect transactions to be atomic and durable - if a transaction fails, no record of it should be stored; if a transaction succeeds, its record should never be lost.

Using the payment service as an implicit key-value store also solves the problem of reliable lookups for news sites. Online payment processors, such as PayPal and Stripe, offer sophisticated APIs on their transaction data. Using this API, an individual can query for the details of a particular transaction, given the proper credentials [11]. This is precisely the functionality that news sites require: given a transaction identifier, a news site wishes to confirm that a payment was made by a particular user for a particular article, and that the payment was of sufficient value. This is all information contained in a transaction record (e.g. in the sender id, payment memo, and payment amount fields, respectively, in PayPal), and accessible via an API call to the appropriate endpoint (e.g. PayPal's /PaymentDetails service) [11].

---

<sup>8</sup>The data store provider could insert transactions for articles the user did not actually purchase, and charge users a small fee for doing so (less than the actual price of the article).

Two major disadvantages exist to having the payment processor double as an implicit transaction data store. Firstly, doing so introduces a static dependency in the system to an external API; if this API changes, our proposed system breaks, unless updates are promptly made to the affected code (which could either be in the client's browser or on the news site's servers). This could be unacceptable for production use cases, in which a major news site must reliably broker payments from thousands of users for site content. Secondly, this design requires both users and news sites to place faith in the payment service. Having entrusted the service to charge a bank account or credit card accurately, it may not be unreasonable for a user to also trust the payment processor to *record* the transaction reliably. But for a news site, using PayPal as the sole transaction data store means delegating all accounting of its own revenue to a non-auditable third-party with its own financial objectives. For large news companies insistent on operational independence, this may not be an acceptable solution, even if the third-party is an established public institution handling nearly \$350 billion in total payment volume each year [14].

2. **Verified email as user id** - Another key design choice is our implementation is the use of a verified email address known to the payment service (PayPal in our implementation) as the universal user id. Portal must retrieve a user's PayPal verified email address on account creation, as the user id is included in a user's public key certificate (obtained by Portal from a certificate authority) and used in the payment verification protocol (see Fig. 3). Two properties of verified email addresses make them suitable for use as a user id in Portal - verified email addresses are 1) globally unique across all users of Portal and 2) not device-specific (such as, for example, MAC addresses), enabling future cross-platform support for the Portal payment protocol.

One slight disadvantage of using a verified email address as a user id is that email addresses are not static. If a user changes the verified email address on record with PayPal, Portal will have to ensure that the user's public key certificate is reissued. Moreover, if an independent transaction data store is used, the user id field will have to be updated on every transaction record for that user. Note that this is not a problem if the payment service itself serves as the transaction data store, as PayPal will update its own records.

3. **Client certificate for authentication** - In the verification protocol outlined in Section 4, we make a fundamental design choice: using client certificates, instead of passwords, for user authentication. The trade-offs between certificate-based and password-based authentication are well-known. Users tend to choose weak passwords that are not resistant to basic brute-force attacks; strong passwords are hard to remember and maintain [19]. The bane of certificate-based authentication, on the other hand, is complexity. Obtaining a certificate from a certificate authority can be a very lengthy process, and certificates themselves are often quite expensive [6, 15]. Using a client certificate, moreover, requires safeguarding a private key. Though private keys can be managed in software (e.g. in Apple KeyChain), the process is opaque to many users, and users run the risk of unknowingly leaking private keys.

Of particular relevance to Portal, password-based authentication entails higher content access latency for users. Under a password-based authentication scheme, users must manually enter in account credentials to purchase a new article, or read a previously purchased story. By introducing friction into the purchase process, this design is likely to discourage impulse purchases of newly discovered news content [20]. Certificates eliminate the manual aspect of authentication, allowing a user with a properly configured browser to prove its identity to a news site without entering (or having to remember) a username and password. Eliminating the upfront costs of account creation with news sites, and the access latency involved in manual login, are key objectives of the micropayments model. These goals motivate our decision to build our payment verification protocol around client-side certificates.

## 6. Evaluation

### 6.1. Usability

Using Portal involves at least three steps of setup, detailed earlier:

1. Creating a Portal account, by providing a verified email address and a password.
2. Linking a PayPal account, by logging in to PayPal from within Portal, and granting Portal the ability to issue payments without further authorization.

### 3. Installing the Portal Chrome extension.

This process poses a significant setup cost, as it involves in particular both account creation and software installation. For users looking to sidestep the time cost of becoming a subscriber, creating an account with Portal may seem equally time-consuming. The practical benefits of using Portal become clearer for users who subscribe to multiple news sites (e.g. The New York Times and The Economist); rather than manage a different account for every subscription, users must only maintain a single, central one with Portal. But for users deciding between purchasing a single subscription and using Portal, the benefits may be primarily economic in nature. Even if equally costly to setup, Portal offers the ability to purchase a wider range of content at a finer level of granularity, allowing users to pay only for content they read.

In practice, using a micropayments system such as Portal involves frequent evaluation of one's willingness to pay for a digital good of small monetary value, such as a news article, a difficult task with a relatively substantial associated cognitive overhead [33, 29]. A number of studies, including a prominent one conducted by AOL in 1996 on flat-rate pricing for Internet access, show that users tend to pay less under metered pricing models, such as Portal's, yet prefer flat-rate plans, because they do not involve the psychological burden of monitoring one's usage [29]. Subscription services, such as Spotify and Netflix, are built on the premise that users will willingly pay a fixed price for unlimited access to content, an idea that many prominent news sites, such as The New York Times, The Wall Street Journal, and The Washington Post, continue to embrace. The recent success of news portal service Blendle, however, which counts 1 million registered and 150,000 paying users worldwide, suggests that a growing number of Internet users may be willing to experiment with an iTunes-like model for online journalism [25].

## 6.2. Privacy and Security

Using the Portal Chrome extension entails enabling the application to read the headers of all HTTP responses, and modify the headers of all HTTP requests made in the browser. At the level of granularity supported by Google Chrome, this translates to granting Portal, on install, the permission

to "read and modify all your data on all websites you visit" [7, 27]. In particular, Chrome does not distinguish between applications that only modify the headers of HTTP requests and responses, such as Portal, and applications that can also modify the *bodies* of HTTP messages, such as ad-blockers and phishing malware. Note that while this permission is requested by many popular Chrome extensions, including Adblock and Adblock Plus [7], some users are bound to find it unacceptably invasive, and opt not to install the extension for reasons of privacy.

Using Portal also involves granting the service the permission to issue payments of up to a certain value at any time from a linked PayPal account. While this is, once again, a capability requested by many popular applications that require payment information, such as Amazon and Uber, users may hesitate to grant this permission to an application with little initial brand reputation. Without the permission to issue payment transactions without explicit authorization, Portal cannot offer a one-click interface, whereby a user can unlock a paywalled article simply by clicking on the Portal Chrome extension. Portal, like other new applications which involve user payments, will thus require its initial users to place trust in its intentions.

Finally, Portal, like many other in-browser applications, will have the capability to maintain data on all news articles a user has read and considered purchasing. The history of news content consumed by a user, while a subset of the data available to search engines and to many other prominent web services which employ third-party tracking, could be highly valuable to third parties, easy to de-anonymize, and potentially incriminating for users. Thus, not treating this data judiciously (i.e. not storing it securely, or intentionally releasing it to third parties) could lead to severe backlash from users.

### **6.3. News site adoption**

To accept payments for news content, news sites will have to run payment verification code distributed by Portal as a server-side module or library. This could potentially be a major barrier to adoption. Given that a news site is willing to experiment with the business model of unbundled, pay-per article pricing, the site will have to assign an engineering team to evaluate the verification



protocol itself, and determine whether the associated access control mechanism meets the company's security standards.<sup>9</sup> The news site will then have to estimate the cost of incorporating external logic into their servers (including installation, deployment, and maintenance expenses), and determine whether the necessary infrastructure is best developed in-house, or imported from Portal. Finally, the news site will have to deploy the new access control mechanism in internal and external testing phases, results from which will determine whether the site will actually choose to adopt the new payment mechanism. This extended period of evaluation and testing, along with the fundamental change in business strategy entailed for news companies, will translate to long sales cycles for Portal, and many opportunities for news sites to exit the potential partnership.

This raises a question for future work - can payment verification for news sites be reduced to a single API call to an external service (such as Portal), which can both 1) verify that a user paid for a requested article, and 2) prove to the news site that the check was actually made? While such a scheme would still involve some work on the part of news sites (namely, adding the API call to server code), it would be a sizable improvement over running three verification checks with each HTTP request. Note, however, that the proof of verification is critical; without it, the news site would in effect be delegating distribution rights over its paid content to a third party, an arrangement that most sites would likely reject.

Another possible approach to payment verification for news sites is to delegate to a third party at the request level, but perform periodic bulk audits of the third party's records. This would serve the purpose of confirming that, over some time period, users did in fact pay for content that was returned to them, while avoiding the overhead of verifying payment with each HTTP request. Building a robust, provably correct system for bulk auditing is an open challenge for subsequent research.

## 6.4. Scalability

Possible scalability bottlenecks that a commercial implementation of Portal could face include:

1. **Payment verification load** - The web servers of prominent news sites such as the New York Times handle millions of page views per day [17]. Given a successful business based on

---

<sup>9</sup>An access control mechanism that includes trusting a third party, such as PayPal, to store transaction data.



micropayments for individual stories, the top news sites would also have to handle high volumes of incoming payment transactions. While the transactions themselves would be directed at merchant accounts owned by news sites, and not their web servers, the burden of matching incoming HTTP requests for content to received payment transactions *would* fall on a news site's server infrastructure. This could add significantly to experienced load times for purchased articles, and throttle the number of paid visitors a news site could potentially support. The success of large online retailers such as Amazon, however, in brokering payments from millions of daily visitors suggests that the problem is not an unsolvable one. Nevertheless, an alternate architecture, which involves decoupling the verification process by outsourcing it to a third-party, such as Portal, via a single API call, could scale better, and is discussed in the section on adoption roadblocks (Section 6.3).

2. **PayPal transaction bandwidth** - PayPal supported 6.1 billion payment transactions in 2016, a 24% increase from 2015 [14]. In January 2016, The New York Times counted 650 million digital page views from 73 million unique visitors, while The Washington Post logged 735 million page views from 70 million uniques [17]. At this scale, if every Times and Post visitor were to purchase 1 article on average per month, PayPal would see an additional 1.7 billion distinct transactions in a given year, or 28% more than it handled in 2016. These numbers suggest that PayPal likely has the capacity to support the increase in transaction volume resulting from the proliferation of a specific new use case for PayPal services, namely, payments for online news, but that the impact could be significant.

Another concern with regards to PayPal transaction bandwidth is peak usage. In 2016, PayPal handled 193 transactions per second on average, and at its peak, handled 450 payments per second on Cyber Monday in 2015. This compares to an average of 3,500 transactions per second, a peak rate of 11,000 transactions per second (near Black Friday), and the purported capability to handle 56,000 transaction messages per second for Visa [1, 35]. A commercial deployment of Portal would have to take into consideration these limits, and evaluate whether PayPal (or another supported payment service) could scale to support the volume of small value payments

that would result from directly monetizing online news.

3. **Client-side state** - As indicated in Section 5.3, Portal must store a record of every payment a user purchases in a private key-value store on the client side. This allows Portal to track previously purchased articles, and avoid reissuing payment transactions for articles a user has already paid for. Clearly, this state scales linearly with the number of articles a user purchases over Portal's lifetime. Even if a user purchases one article every day for five years, this amounts to less than 2000 key-value records, each of which is unlikely to exceed 1 KB in size (the key is an article URL, the value a fixed-length transaction id). Thus client-side state is unlikely to be a scalability bottleneck for Portal. Even if it does become one, garbage collecting old data (i.e. storing it only on Portal's server, not in the user's browser) offers a viable solution.

## 7. Deployment and Adoption

### 7.1. Launch

Portal will face the adoption issues of a two-sided market. For the service to hold initial value for users, payments to some set of popular subscription news sites will have to be supported from the onset. For news sites to adopt the new, unbundled content distribution and monetization model entailed in Portal, along with the associated payment protocol, Portal will have to demonstrate the potential to reach a significant number of a site's readers, and serve as a viable source of revenue.

One sensible launch strategy may be to first target regional news sites (e.g. The Chicago Tribune, The Baltimore Sun), which have cultivated large, loyal readership bases, but have generally struggled to convert these readers into paying, digital subscribers [24]. Another category of sites that might serve as strong initial partners include technology and financial news sites, such as Wired and The Economist, which currently sell paid membership and subscription offerings, and cater to niche, technologically-savvy audiences. The Economist, in particular, is charting an aggressive course to increase profits via wider digital penetration, and has demonstrated a concerted willingness to experiment with pricing and product offerings [34]. These include various packagings of its core subscription offering, and a daily news application, Espresso, priced at \$3.30 a month, for users

not interested in purchasing a full subscription at all [34]. Used by 40% of Economist subscribers, and claiming 200,000 weekly readers, Espresso demonstrates that opportunities do exist for news sites to expand revenue via a lower price point content offering, without cannibalizing their existing subscription base [34].

## **7.2. Initial Audience**

Portal would begin by targeting users who have considered becoming subscribers in the past, but have chosen not to, due to inertia or unwillingness to manage another monthly bill. Many current subscribers are high-income professionals who place a high value on premium news content (e.g. enough to pay \$335-610 a year for a Financial Time subscription [9]). This would be Portal’s initial, target demographic. A reasonable next step may be to expand to consumers of niche news content (e.g. technology and fashion), who may be willing to pay for long-form, journalistic pieces, on a case-by-case basis, in their area of interest.

## **7.3. Product Discovery**

Paywalls today serve the purpose of pointing users to the various subscription packages offered by a news site (see Figs. 8, 9). An organic way for news sites to refer users to Portal would be to list Portal as a second, pay-per-article payment option on their paywalls. To offset the initial barriers to using Portal, which include creating an account, downloading a Chrome extension, and linking a PayPal account or credit card, Portal could subsidize the first set of articles purchased by a user (i.e. offer  $x$  free articles). This would allow users to defer the account creation and linking steps until they have used Portal, and experienced its utility.

# **8. Future Work**

## **8.1. Payment Mechanisms**

In our prototype, we implemented support for purchases with PayPal, but a commercial implementation of Portal may wish to support other payment mechanisms, such as credit cards and Bitcoin.

Figure 8: The New York Times Paywall [12]

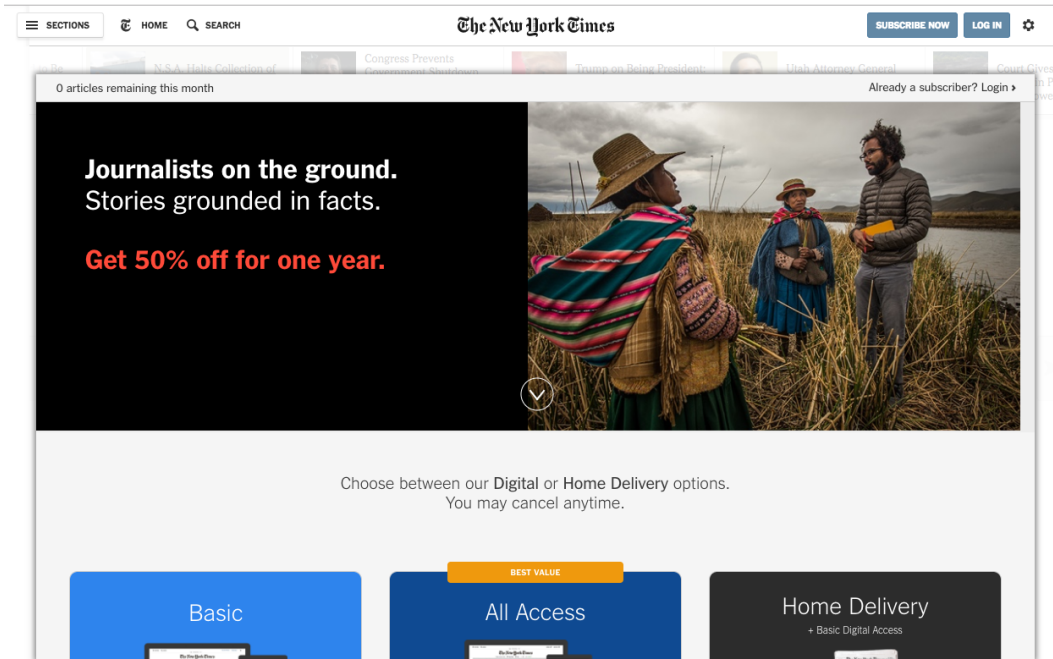





Figure 9: The New York Times Subscription Offerings [13]

Basic	BEST VALUE All Access	Print Delivery + All Access
 <b>\$1.00</b> <del>\$6.25</del> /week As long as you're a student	 <b>\$1.50</b> <del>\$6.25</del> /week As long as you're a student	 As low as <b>\$4.95</b> <del>\$9.90</del> /week* Get 50% off
<b>GET DIGITAL ACCESS</b>	<b>GET ALL ACCESS</b>	<b>GET PRINT DELIVERY</b>
<b>Basic Digital Access Includes:</b>  Access to NYTimes.com and all New York Times apps.  Unlimited article access, anytime, anywhere.  Access to the complete Times archives from 1851.	<b>Basic Digital Access Features</b>  + <b>Times Insider Access</b>  Access to exclusive, behind-the-scenes stories, photos and videos from our journalists inside the newsroom.  Attend live panel discussions and special events hosted by Times editors and journalists.  Access to a weekly "Inside The Times" podcast series and a collection of 120+ e-books — collections of our most important stories — curated by Times editors.	<b>Basic Digital Access Features</b>  + <b>Times Insider Access</b>  + <b>Print Delivery</b>  Convenient and reliable delivery. Guaranteed.  Customized delivery options such as Sunday only, Fri.-Sun., weekday delivery or daily delivery.  Includes weekly Sunday magazine and monthly T Magazine.

With over 190 million active user accounts worldwide, PayPal has established itself as a standard mode of website commerce, but many users still prefer to use credit cards [14]. On mobile devices, native services, such as Apple Pay and Google’s Android Pay, offer more natural interfaces for issuing payments, and are strongly gaining ground over incumbents such as PayPal [31].

The semantics of Bitcoin transactions, which can be unambiguously identified in a public ledger by both the client and the seller, make them particularly well-suited to payments for digital goods based on the HTTP 402 Protocol. Bitcoin offers a natural mechanism for payment verification. Issuing a Bitcoin transaction requires a signature from the private key associated with the sender’s Bitcoin address; a signature from the same private key on a request id assigned by the news site would cryptographically link the HTTP request for the article with the Bitcoin transaction issued to pay for it. This is precisely the goal of the “proof-of-payment” provided in the verification protocol described in Section 4. Using Bitcoin eliminates the need for the user to supply a public key certificate to the news site with each request (and the need for a Certificate Authority to issue one for each Portal user). Instead, given the private-public key pair  $(SK, PK)$  associated with the Bitcoin address  $A$  from which the transaction is issued, the user must provide only: 1) a signature, made with  $SK$ , on the assigned request id, and 2) the public key  $PK$ . Note that a Bitcoin address  $A$  is simply an encoded hash of the associated public key  $PK$ . Using the provided public key  $PK$ , the news site can 1) verify the user’s signature on  $r$  and 2) check that payment of sufficient value was in fact received from an address  $A' = h(PK)$ .

At the time of writing, Bitcoin does not have sufficient adoption, in our opinion, for it to serve as Portal’s primary payment mechanism. Bitcoin also poses two technical challenges that could stymie an implementation of micropayments built on top of it: 1) Bitcoin transactions take ten minutes on average to appear in the public ledger, a facet of how Bitcoin mining is globally calibrated (though this problem can be sidestepped by introducing an escrow protocol involving an intermediary third party), and 2) the Bitcoin network is severely bandwidth-challenged (it can only support 7 transactions per second at the current Bitcoin block size), a more serious, political issue [3]. Bitcoin, however, offers some definitive benefits over standard payment methods in the context of

this application - namely, small value transactions for digital goods - with its low transaction fees, programmatic nature, and auditability. These advantages lend credence to implementing future support for Bitcoin, along with other payment mechanisms, including credit cards and flagship mobile checkout services (e.g. Apple Pay and Android Pay).

## **8.2. Mobile Devices**

While Portal enables the purchase of news articles from desktop browsers, another common use case for the service would likely involve paying for and reading content from mobile devices. Most major browsers do not currently offer browser extensions for mobile devices, so Portal's desktop solution would not translate as is to mobile. Instead, support for mobile access could come in one of the following forms:

1. **Portal mobile app** - A dedicated mobile app for Portal would allow users to create/log in to a Portal account, link a payment method, and issue payments to news sites for content. The app could serve solely as a payment service, integrating closely with a native browser (e.g. Chrome, Safari) on a user's phone. Under this model, users would discover content on their favorite browser, switch to Portal to issue payment to the providing news site, and then switch back to the browser to read the unlocked article. Such a design could pose serious user experience problems. Alternatively, the app could function as a central platform for reading paid news content. This would stipulate that the app serve as an effective mobile browser, significantly increasing its technical complexity and raising the level of buy-in required from users.
2. **Single sign-on access** - Portal could offer access to articles purchased from a desktop browser through a single sign-on interface on news sites. This would require users to be logged in to a Portal account on their mobile browser, and then use Portal to log in to news sites. Such a scheme would not support the purchase of *new* articles on a mobile phone, and would require buy-in from news sites (which would have to accept Portal as an identity provider).
3. **Webpage plugin** - In addition to supporting single sign-on, news sites could add Portal plugins to the mobile versions of articles. This plugin would serve the same purpose as the Portal browser

extension on desktop devices; clicking on it would trigger an API call to the appropriate payment service (e.g. PayPal), and a money transfer to the hosting news site. Just as with single sign-on, this will require a user to be logged in to Portal on their mobile browser, to enable the plugin to execute an authenticated API call. News sites would be required to load an executable script from Portal on each article, but this is common practice (consider: social plugins, Google Analytics scripts, third-party tracking devices).

Coupled with single sign-on access, the webpage plugin approach may be the lightest-weight solution to supporting both content purchase, and paid content access, from mobile devices.

## **9. Conclusion**

In this paper, we presented Portal, a payment protocol and software system underlying a new monetization model for online journalism centered on small payments for individual stories. We began by motivating our departure from the paid subscription model, addressing three key shortcomings - inflexible pricing, cognitive overhead, and high upfront time costs - that deter users from subscribing, even to top news sites. In its place, we presented an alternate monetization system for subscription-only and paid membership news sites characterized by pay-per-article pricing, site-agnostic payments, and one-click payment flows, each of which address a particular challenge of the paid subscription model. We then discussed two related services, Blendle and PayPal, before outlining our core contribution: a payment protocol for claiming a digital good from a content provider over HTTP, after issuing payment over a parallel channel (e.g. PayPal, Stripe, credit card network, Bitcoin). We described the entailed payment verification problem, and presented a solution based in standard public key cryptography which involves the provision, by users, of 1) a valid public key certificate and 2) a digital signature on a server-assigned request id. We concluded the section by analyzing a broad category of attacks that the verification procedure protects against, in the process uncovering some exploits that are still possible.

The second part of our paper addressed our particular implementation of the outlined payment system. Portal is composed of an account service, browser extension, and server-side payment

verification code. We addressed both the setup and operation of these components, and three key design choices that underlie our system: the use of PayPal as a transaction data store, verified email addresses as universal user ids, and client-side certificates for user authentication. Following this, we evaluated both our architecture and implementation on a number of different criteria: 1) usability, including set up costs and the psychology of micropayments, 2) privacy and security, including permissions, and the handling of payment credentials and user data, 3) adoption challenges posed by modifications to server-side infrastructure, and 4) potential scalability bottlenecks, including verification load, transaction volume, and client-side state. We then outlined a launch strategy, identified an initial audience, and addressed product discovery in a section on deployment and adoption. We concluded with an in-depth treatment of two possible areas for future work – support for other payment mechanisms, including credit cards and Bitcoin, and the problem of enabling content purchase, and paid content access, on mobile devices.

Looking forward, we plan to validate our proposed system with the business development and technology teams at the New York Times, release a first version of the Portal Chrome extension to the Chrome Web Store, and beta test the system with a prototype news site and a select group of invited users. Feedback from news sites and our initial cohort of testers will inform the next steps we take with regards to the development of Portal, and its deployment to users.

## References

- [1] “Visa inc. at a glance,” <https://usa.visa.com/dam/VCOM/download/corporate/media/visa-fact-sheet-Jun2015.pdf>, June 2015.
- [2] “Los angeles times - digital fast facts,” <http://mediakit.latimes.com/Media/LosAngelesTimesMediaKit/Toolkit/Digital%20Fast%20Facts.pdf>, March 2016.
- [3] “Scalability - bitcoin wiki,” <https://en.bitcoin.it/wiki/Scalability>, December 2016.
- [4] “Us ad blocking to jump by double digits this year - emarketer,” <https://www.emarketer.com/Article/US-Ad-Blocking-Jump-by-Double-Digits-This-Year/1014111>, June 2016.
- [5] “Adaptive payments api reference - paypal developer,” <https://developer.paypal.com/docs/classic/adaptive-payments/integration-guide/APIIntro/>, 2017.
- [6] “Certificates and public keys,” [https://msdn.microsoft.com/en-us/library/windows/desktop/aa376502\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa376502(v=vs.85).aspx), 2017.
- [7] “Chrome web store,” <https://chrome.google.com/webstore/category/extensions/>, 2017.
- [8] “Copyright notice - the new york times,” <https://www.nytimes.com/content/help/rights/copyright/copyright-notice.html>, 2017.
- [9] “The financial times - make the right connections,” [https://sub.ft.com/spa2\\_5/](https://sub.ft.com/spa2_5/), May 2017.
- [10] “Financial times - subscribe to read,” <https://www.ft.com/signup?offerId=c8ad55e6-ba74-fea0-f9da-a4546ae2ee23/>, May 2017.



- [11] “Introducing adaptive payments - paypal developer,” <https://developer.paypal.com/docs/classic/api/adaptive-payments/>, 2017.
- [12] “The new york times,” <https://www.nytimes.com/>, 2017.
- [13] “The new york times digital subscription at the academic rate,” <http://www.nytimes.com/subscriptions/edu/lp8LQFK.html>, May 2017.
- [14] “Paypal reports fourth quarter and full year 2016 results,” <https://investor.paypal-corp.com/releasedetail.cfm?releaseid=1009339>, Jan 2017.
- [15] “Ssl and tsl certificates by symantec,” <https://www.websecurity.symantec.com/ssl-certificate>, 2017.
- [16] “What is paypal? learn more about paypal’s unique features,” <https://www.paypal.com/en/webapps/mpp/paypal-popup>, 2017.
- [17] J. Barr, “The new york times pulls back ahead of the washington post for unique visitors,” <http://adage.com/article/media/york-times-pulls-back-ahead-washington-post/302720/>, Feb 2016.
- [18] A. Bhattacharya, “Would you pay for journalism if you could get your money back on clickbait?” <http://www.theverge.com/2016/3/23/11286072/blendle-micropayments-journalism-money-back-clickbait>, March 2016.
- [19] J. Bonneau, “The quest to replace passwords: A framework for comparative evaluation of web authentication schemes,” in *IEEE Security and Privacy*, 2012.
- [20] K. Eaton, “How one second could cost amazon 1.6 billion in sales,” <https://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales>, Mar 2012.
- [21] S. Embers, “New york times co. reports loss as digital subscriptions grow,” <https://www.nytimes.com/2016/05/04/business/media/new-york-times-co-q1-earnings.html>, May 2016.
- [22] G. Ferenstein, “The psychology behind the new york times paywall,” <http://www.fastcompany.com/1740113/psychology-behind-new-york-times-paywall>, March 2011.
- [23] J. Greenberg, “The new york times says it has more subscribers than ever,” <https://www.wired.com/2015/10/new-york-times-subscribers-ever/>, October 2015.
- [24] D. Harwell, “Gannett’s dropped tronc bid again highlights struggles for the newspaper industry,” <https://www.washingtonpost.com/news/business/wp/2016/11/01/gannetts-dropped-tronc-bid-again-highlights-struggles-for-americas-newspaper-industry>, November 2016.
- [25] S. Hepworth, “Blendle reaches 1 million users, but is it here to stay?” [https://www.cjr.org/analysis/blendle\\_signups\\_aggregator\\_micropayments.php](https://www.cjr.org/analysis/blendle_signups_aggregator_micropayments.php), August 2016.
- [26] T. Julian, “The 21 bitrequests library (two1.bitrequests),” <https://21.co/learn/21-lib-bitrequests/>, 2016.
- [27] T. Klosowski, “Why do chrome extensions need to access all my data?” <http://lifehacker.com/5990769/why-do-chrome-extensions-need-to-access-all-my-data>, March 2013.
- [28] K. Lu and J. Holcomb, “Digital news revenue: Fact sheet,” <http://www.journalism.org/2016/06/15/digital-news-revenue-fact-sheet/>, June 2016.
- [29] A. Odlyzko, “The case against micropayments,” in *Financial Cryptography*, 2003.
- [30] J. W. Peters, “The times announces digital subscription plan,” <http://www.nytimes.com/2011/03/18/business/media/18times.html>, March 2011.
- [31] L. Rao, “How paypal plans to get back on top in digital payments,” <http://fortune.com/paypal-fortune-500-digital-payments/>, June 2016.
- [32] M. Scott, “Study of ad-blocking software suggests wide use,” <https://bits.blogs.nytimes.com/2015/08/10/study-of-ad-blocking-software-suggests-wide-use/>, August 2015.
- [33] C. Shirky, “The case against micropayments,” <http://archive.oreilly.com/pub/a/p2p/2000/12/19/micropayments.html>, December 2000.
- [34] L. Southern, “The economist plans to double circulation profits in 5 years,” <https://digiday.com/uk/economist-aims-double-circulation-profits-5-years/>, February 2016.
- [35] M. Trillo, “Visa transactions hit peak on dec. 23,” <http://www.visa.com/blogarchives/us/2011/01/12/visa-transactions-hit-peak-on-dec-23/index.html>, Jan 2011.