A Protocol for Privately Reporting Ad Impressions at Scale

Matthew Green
Johns Hopkins University
Baltimore,MD
mgreen@cs.jhu.edu

Watson Ladd
UC Berkeley
Berkeley, CA
wbl@math.berkeley.edu

Ian Miers
Johns Hopkins University
Baltimore,MD
imiers@cs.jhu.edu

Abstract

We present a protocol to enable privacy preserving advertising reporting at scale. Unlike previous systems, our work scales to millions of users and tens of thousands of distinct ads. Our approach builds on the homomorphic encryption approach proposed by Adnostic [42], but uses new cryptographic proof techniques to efficiently report billions of ad impressions a day using an additively homomorphic voting schemes. Most importantly, our protocol scales without imposing high loads on trusted third parties. Finally, we investigate a cost effective method to privately deliver ads with computational private information retrieval.

1. INTRODUCTION

Since the inception of the web, users have debated the appropriate means for funding the creation of web content. In recent years, however, the debate has become more contentious. This due in part to an increase in the sophistication of targeted advertising [16], and in part to a corresponding increase in the use of client-side ad blocking and anti-tracking software [7]. These trends may be related. According to a U.S. Department of Commerce study [23], nearly 23% of households identified data collection by online services as a "major concern", and 35% of these users report that they have avoided online activity as a result.

A number of technical solutions have been offered to improve user confidence in the privacy of the web. These include micropayment systems that provide alternatives to advertising [39, 32], as well as new advertising systems that offer an improved balance of privacy and functionality [1].

In the latter category, a promising approach to solving these problems is to deploy *privacy preserving ad delivery* systems that limit advertisers' ability to track users, while simultaneously providing user-specific and meaningful advertising. In a privacy-preserving advertising system, a local client (*e.g.,* the browser) monitors the user's web history and selects display ads from a set of available advertisements that have been either downloaded as part of the browser itself or obtained through PIR [8]. The client then displays the ads appropriately within a web page, without transmitting user-identifiable information to the website or advertising network. The technical challenge for privacy preserving advertising is ensuring that the client's ad preferences are not exposed to the ad provider or website, while at the same time guaranteeing that the website receives compensation for aggregate ad displays and "conversions" (clicks on a displayed ad).

A number of private ad network architectures have been proposed in the research literature [27, 11, 42]. These generally take one of two approaches: either they anonymize the user when she submits in clear an ad impression, or they obscure which ad was viewed when an identifiable user submits ad impressions. Systems in the former category, such as [27], use mixnets, trusted hardware on the provider back-end [11], or anonymity networks operated by non-collaborating parties. Systems in the latter category, e.g. [42, 24], use cryptographic voting protocols: each ad is viewed as a candidate in an election and tabulated ballots reveal the total number of impressions each ad received — *without* revealing what each individual ballot "voted" for.

Advertising at scale. In this work we focus on an aspect of the problem that has been largely ignored in the previous work: the problem of deploying privacy-preserving advertising *at scale.* The previous work in this area considers only small user sets, or relies on (not yet widely available) trusted hardware at the datacenter in order to support large volumes of advertising. By contrast, our goal is to use current technology to construct a privacy-preserving ad system for a platform that scales to a user base consisting of *hundreds of millions of users.*

This is not a theoretical problem. Our work is motivated by a collaboration with one of the major browser vendors, which is investigating the possibility of deploying a privacy-preserving advertising system for ads generated in the New Tab screen of the browser. This considerably simplifies the system, as we need only record which ads were viewed and *not* attribute where the ad was displayed: all ad revenue for successfully displaying an ad is paid to the browser vendor.

Despite this simplified setting, the problem is quite challenging: discussions with the vendor indicated that such a system would need to support 100 million daily users and 2 to 4 *billion* individual ad impressions per day. More interestingly, the computational costs of the scheme are required to be less than 1% of the total revenue obtained, expressed in the form of *cost-per-thousand-impression* (CPM). In ad-

CCS'16, October 24 - 28, 2016, Vienna, Austria
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4139-4/16/10. . . $15.00
DOI: http://dx.doi.org/10.1145/2976749.2978407

Scheme	Cost Per Thousand Impressions (CPM)			Impression size			Security Provided	
	4,000 ads	32,000 ads	64,000 ads	4,000 ads	32,000	64,000 ads	under count	over count
Paillier [36]	$0.01	$0.08	$0.17	1.3 MB	10 MB	20 MB	Yes	No
Adnostic [42]	$0.09	$0.73	$1.46	8 MB	64 MB	128 MB	Yes	Yes
AdScale §5	$0.003	$0.01	$0.02	188 KB	1.3 MB	2 MB	Yes	Yes

Table 1: AdScale performance comparison. Note that numbers for Paillier are from an optimized version produced for this paper. See §1.1

dition the allowable bandwidth consumption on each client is relatively limited.

An analysis of the literature indicates that existing private advertising techniques are unlikely to scale realistically to this use case, either due to the need to construct an enormous new anonymity network (which would exceed the total user count of Tor by an order of magnitude [6]), or because of the cost of reporting the expected advertising volume. Moreover, scaling some of the existing advertising systems requires large-scale deployment of trusted hardware (*e.g.,* [11]) or else the availability of trusted third parties that can operate highly-available back end systems.[1]

Our contribution. To address these unique scaling requirements, we introduce a new and efficient technique for reporting ad impressions based on the cryptographic voting paradigm. Unlike previous work, our solutions scale to meet the necessary requirements, *without* the need for trusted hardware or unrealistic effort from trusted parties. The key advance in our system is a new and dramatically more efficient cryptographic technique that reduces the overhead of reporting ad impressions by orders of magnitude when compared to the existing literature [42, 36] (see Table 1), while providing improved reporting functionality. Our system improves on the cryptographic voting model previously used in systems such as Adnostic [42], but reduces the bandwidtch of proving the correctness of each ad report from $O(N)$ to $O(log\ N)$ when dealing with N possible advertisements. While we concentrate on advertising in this work, we note that our techniques may also have other applications.

To validate our techniques, we implement our scheme with extensive optimizations, and show that — when considering large numbers of possible advertisements — it improves computational performance by an order of magnitude when compared to the state of the art in Paillier-based voting systems [36], and three orders of magnitude when compared to previous Elgamal-based solutions [42]. More importantly, we analyze the financial costs of our approach in terms of computational and bandwidth overhead per impression, and demonstrate that the computational costs of the scheme fall below 1% of the estimated revenue obtained in current advertising systems. Finally, we argue that with appropriate sharding of ad campaigns, our techniques can scale to even larger campaigns.

While our work in this paper is largely concerned with reporting advertising impressions, we additionally investigate the problem of delivering advertising content. Specifically, we examine the costs of various delivery strategies, including pre-positioning and recent results in computational Private Information Retrieval (PIR) [8]. Interestingly, these results show that even computational PIR schemes [8] are cost effective if only *server-side* computational and bandwidth costs are considered — costing less than 2% of CPM for delivering 1000 ads.

1.1 Existing techniques

Each of the anonymous advertising systems we examined falls into one of three categories. Systems based on Mixnets, such as the proposal of Juels [27], rely on mutually non-colluding third parties to cryptographically anonymize reports on ad statistics. Voting systems such as Adnostic [42] transmit encrypted vectors that contain a 1 in each position corresponding to a viewed advertisement, and a 0 in every other position, allowing an honest-but-curious central party to homomorphically sum the ciphertexts into a single aggregate ciphertext that can be decrypted by a trusted party. Finally, hardware-based approaches such as Obliviad [11] use back-end servers with specialized co-processors to hide user data from the ad network. We now briefly present some comments on the scalability of these existing techniques:

Scalability of anonymity networks. Mixnets [27] and onion routing networks (*e.g.,* Tor) seem unlikely to meet our requirements, due to the number of trusted parties required: for these systems to be secure, at least one party who mixes or routes each message must be honest. This means identifying a collection of non-collaborating and *highly available* nodes whose number scales appropriately to handle the bandwidth requirements of the network. To illustrate the challenges, we analyze the potential load on the Tor network in §A. Independent of advertising, a long line of works have considered scalable anonymity networks. These run the range from improvements to Tor [41], to alternatives [31, 33, 35], to a line of works on scalable mixnets [20, 21], and dining cryptographer networks [17, 43, 18]. To the best of our knowledge none of these proposals can operate at the scale we require — either due to technical limitations, or simply because a deployment would require an implausible number of non-cooperating parties.

Limitations of hardware-based approaches. Hardware-based systems such as [11] seem like a promising solution. However, these do not currently seem feasible without increasing the cost of ad delivery beyond the point of profitability.[2]

[1]While a reliance on trusted parties seems reasonable, our goals require that parties have realistic capabilities and operational expenses. This is particularly important at the scale we are concerned with, since the advertising network cannot compensate the third party for large operational expenses without potentially creating a conflict of interest.

[2]The commercial availability of platforms such as ARM TrustZone [9] and Intel's SGX [10] raises the possibility that such deployments may be feasible in the future, but widespread availability of these technologies in commodity cloud computing platforms remains some years in the future.

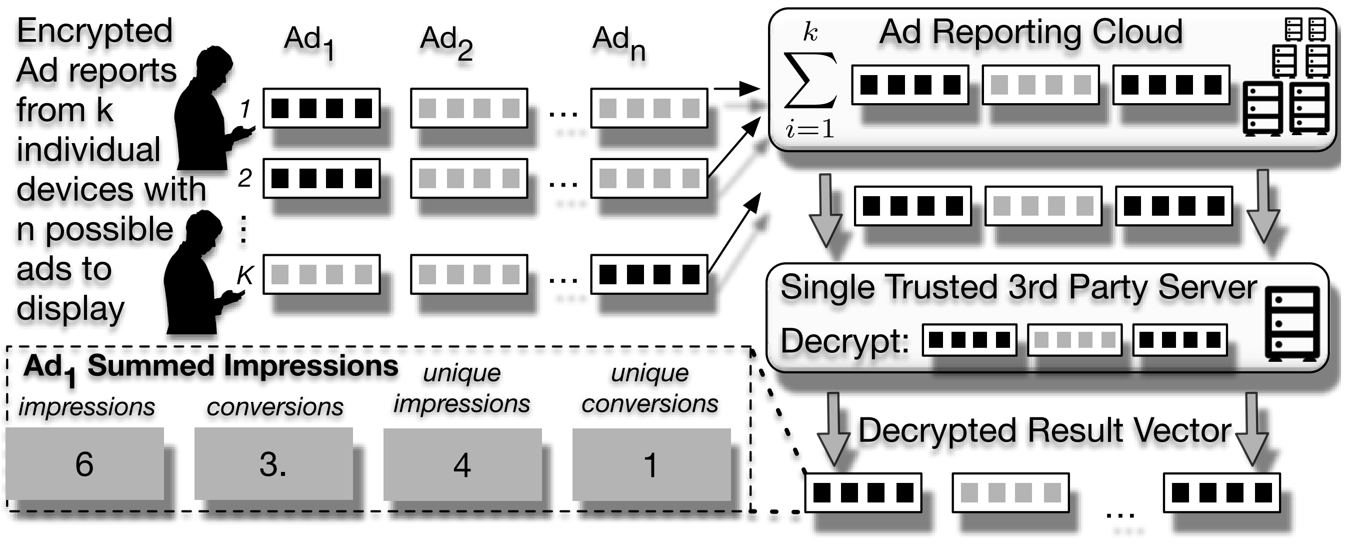


Figure 1: AdScale architecture. Clients select ads based on device only profiles, display ads, and then report impressions for those ads. Only aggregated impressions are ever revealed. The ads an individual user views therefor the profile built of that user, remain private except to the trusted third party (who's key can be split of multiple independent parties if need be).

Costs of existing voting systems. Unfortunately, the existing voting systems tend to scale poorly with the number of ad campaigns (see Table 1). The challenge is typically the need to *prove* (in zero knowledge) that the vectors are correctly structured. Without such a proof, an adversarial user can insert a large positive or negative integer into a ballot, thus substantially affecting total ad count. These proofs substantially increase the bandwidth cost of each vector, and induce costs on the data center. Protocols such as Adnostic [42] induce an $O(N)$ bandwidth cost for this proof. Based on an experimental implementation, this proof and ciphertext requires 8MB of data to report a modest 4,000 element ballot, and a proof verification rate of two ciphertext vectors per second per core on a high-end Intel processor (even after extensive software optimizations and batch processing). Moreover, systems such as Adnostic do not include auxiliary data such as *conversions* (whether the ad was clicked), and if those conversions or impressions were *unique*.[3]

Paillier-based systems [24] also fail to meet our requirements, though for a different reason. In these schemes the size of a Paillier public key size grows linearly with the number of ads. Since the computational costs of encryption increase quadratically, even with careful optimization we are forced to cap the size of each vector and use multiple vectors to report a single impression. But the proof of correctness only applies *per* vector, and provides only partial guarantees about the entire ballot. In particular, while this prevents *undercounting* because it guarantees that ever value in the set of vector is either 0 or 1 (and not, e.g. negative), an attack can *overcount* impressions by setting 1 bit high in each vector in an impression.

1.2 Improved correctness proofs for ballots

The main contribution of this work is a new and asymptotically more efficient approach to proving the correctness of an encrypted ballot. A vector of ciphertexts (a ballot) contains the encryption of a bit vector that indicates which ads the client has viewed. These ballots are submitted by the client and homomorphically summed by the ad reporting network. The single summed ciphertext may then be decrypted by a trusted third party who holds the decryption key. The results are returned to the ad network. This ensures that despite knowing the client's identity, the ad network does not learn what ads the client viewed. In essence, this construction is a simple cryptographic voting scheme: candidates are ads, ballots are submitted by clients to report which ad they review, and the results determine how much the advertiser pays. As with cryptographic voting, we must prove the ballot is correct. However, in this case, we must do so for ballots containing thousands of "candidates" — far beyond the scale seen in any real election.

Our approach to proving ballot correctness relies on an efficient non-interactive proof of knowledge (NIZKPoK) for the following informal statement:

> *Given a vector of Elgamal ciphertexts $C_1, \ldots, C_N$, there exists an index j such that C_j encrypts plaintext 1, while each of the remaining ciphertexts encrypt 0.*

Let g be a generator of a cyclic group $\mathbb{G}$ of prime order p. The naive approach to proving the above statement is to encode each plaintext m_i (for $m_i \in \{0,1\}$) as a group element g^{m_i}. We then encrypt this group element using Elgamal encryption in $\mathbb{G}$, and prove that the product $C^* = \prod_{i=1}^{N} C_i$ decrypts to g^1. This is easily verified since the sum of the plaintexts can be computed using the homomorphic properties of the encryption scheme. Of course, this proof is insufficient by itself, as it is simple to construct an invalid message vector such that the sum will overflow and wrap

[3]Including this data results in a 4-fold cost increase.

around to 1. The standard defense against this attack (exemplified by Cramer *et al.* [19]) explains the high cost of the proof: the prover must also issue a separate NIZK proving that each ciphertext encrypts either 0 or 1.

When implemented using standard techniques [19, 42] over 256-bit elliptic curve groups, this method results in a proof size of approximately 200 bytes per encrypted bit. Even if the proof verification is made amenable to batched verification, this approach produces unacceptable performance at scale. Moreover, in a practical system each advertisement may require several bits of reporting information due to the need to encode several different facts about the advertising interaction, and thus a relatively small number of advertisements quickly consumes megabytes of bandwidth per impression. This inordinate bandwidth consumption makes mobile advertising infeasible with this scheme.

To address this weakness, we propose a novel proof technique that reduces the size of the proof from $O(N)$ to $O(log\ N)$ while dramatically reducing verification time. Our approach uses ideas from the recent work of Groth and Kohlweiss [25], who propose an efficient proof of membership for a committed value in a list of values. We now describe the intuition of our approach.

We first force the prover to commit to the ciphertext vector $C_1, \ldots, C_N$ where each ciphertext is the Elgamal encryption of an encoded message in $\mathbb{G}$. Following this commitment, an honest verifier generates a vector of random elements $a_1, \ldots a_N \in \mathbb{Z}_p$, and causes the prover to homomorphically compute the sum $\sum_{i=1}^{N} b_i a_i$ using the ciphertexts.[4] If the prover is honest, then the resulting sum of is equal to a_j for some $j \in [1, N]$. Indeed, thus far the technique is reminiscent of a technique described by Henry *et al.* [26] in CCS 2011. Critically, and unlike the proof of Henry *et al.*, which required a proof of complexity $O(N)$, we are able to prove that the sum is correct using an efficient log-sized membership proof by Groth and Kohlweiss [25]. This avoids the $O(N)$ proof size from the previous approaches, and dramatically reduces the concrete cost of the protocol.

With this technique in hand, we are then able to make considerable further optimizations to both to the protocol and the implementation. First, we generalize the statement to show that the vector of positive numbers sums to *any* specified integer, at a modest additional cost. This extension enables multiple ad impressions to be reported in a single vector of ciphertexts, thus reducing bandwidth demands and computational costs. Second, we extend the basic homomorphic encryption scheme to store multiple bits in a single Elgamal ciphertext, resulting in a 4 fold decrease in bandwidth. Finally, we reduce the ciphertext size to a single group element by employing a distinct public key per ciphertext. Finally, to improve verification and decryption time, we apply a sequence of implementation optimizations.

1.3 Outline of this paper

The remainder of this paper proceeds as follows. In §2 we provide some background on the reporting requirements of online advertising systems. In §3 we describe AdScale, a design for an advertising scheme based on our new techniques. In §4 we provide technical preliminaries, and in §5 we describe the cryptographic voting protocol that is the main contribution of this work. In §6 we present experimental results for our system.

2. BACKGROUND ON ADVERTISING

Many websites are funded through the display of advertisements on their webpages. Advertisements online are increasingly paid for by clicks, with 66% of revenues in 2015 deriving from non-display advertising [38]. Systems for privacy preserving advertising must support a wide range of currently existing revenue models, from simple display ads paid for by the number of users who see them, to complex rewards for completed orders on linked e-commerce sites. They must also permit demographic breakdowns of who is responding to an advertisement and demographic targeting, along with exact and relatively quick tabulation of how many users saw an advertisement.

Advertisements are frequently targeted to users based on user interests and behavior. Search history and website history can indicate intent: a user who searches for “new fords” and then checks out the Blue Book for the value of a 1995 Mustang is considered more likely to click on an advertisement for cheap Ford cars. The exact details of how ads are targeted is not the primary consideration of this work, and so we will not discuss it further, beyond noting that in our model, ads are selected by a program local to the user’s machine that has access to a detailed profile of the user. Advertisements may have various payout models. At the simplest extreme, consider a simple display ad for a car dealership: the site that presents this ad gets paid every time the ad is shown. More complex is a clickthrough ad for the dealership, where the dealer pays for each click and ensuing visit to their website. More complex still, the dealer might have no interest in repeat visitors and thus wishes to pay only for unique clicks. At the far end of the spectrum, the dealer might specify rewards for clicks that lead to the user eventually conversing with a representative or even completing a purchase. Although we are not concerned with user privacy after she interacts with the dealer’s website, all of these details need to be recorded at the client in order to ensure that the website is properly compensated for the resulting conversions.

To address these complexities, a behavioral advertising system must be capable of reporting on a number of outcomes. These are expressed as a vector of *counters* for each ad. These counters can measure impressions (how many times the ad is displayed), conversions (how many times its clicked), and unique impressions and conversions (how many unique users viewed/interacted with the ad). Each report may also record language and quantized GeoIP data, thus permitting demographic analysis and confirmation that the advertisement is targeting the right people: there is little point in a California auto dealer displaying advertisements in Little Rock. For complex conversions such as contacting a representative we can provide the advertiser with HTML tags that will trigger a report, thus enabling a count of these conversions.

Currently most behavioral advertising relies on cookies which are periodically destroyed, which produces high error rates in advertising delivery. Because our counters cannot function as user identifiers, they can persist for longer then cookies without the same privacy concerns as cookies, and so will be more accurate. We also can report on exactly which elements of a profile triggered an ad display: was it

[4]Recall, while we can only add ciphertexts in partially homomorphic encryption, we can multiply by public constants.

the user's interest of cars as ascertained from sites frequently visited, or was it a recent search? This information can help advertisers determine what they should target their ads to. As a result of needing to gather this data for each advertisement, we have to multiply the costs of all schemes by a factor of eight as each advertisement is likely to require at least four and usually eight separate counters to obtain enough information.

Advertising fraud. We now briefly consider the issue of fraud. The first type, commonly called click fraud, involves websites defrauding customers by submitting reports for impressions that did not happen. We note that in AdScale, users are not anonymous and as such, many of the standard techniques for preventing click fraud, such as rate limiting, can be applied.

A more serious issue is advertisers defrauding websites: malicious advertisers could submit encryptions of negative numbers to obtain free displays or conversions, or positive numbers to use up the budgets of rivals and end campaigns early. Because we include a proof limiting the encrypted values, standard anti-fraud solutions such as rate limiting and correlating GeoIP and language information are applicable.[5]

3. DESCRIPTION OF OUR SYSTEM

We now describe AdScale, our architecture for a privacy-preserving advertising platform.

Participants and operation. In AdScale there are three types of party: a *client* which picks ads to display and reports on them, an *ad server* which aggregates reports and serves ads, and a separate *trusted party* who holds decryption keys and periodically decrypts summary reports.

To use the system, the client selects an ad to display and downloads the *creative* (the picture or text shown to the user) through either a cPIR scheme, a pre-positioned database, or a Content Distibution Network (CDN) file containing multiple creatives. These alternatives have various costs and privacy implications that we discuss in §6. The client uses now encrypts a report of this display using the homomorphic encryption scheme. This includes multiple bits of information, including: whether the ad was displayed, whether the ad was clicked, and auxiliary information about how it was chosen. This requires sending between 4-8 separate bit vectors: in our measurements we assume the worst case of 8. Each of these vectors includes one element for every possible advertisement. The client also transmits demographic information such as language and geoIP information. This information is required for evaluating the impact of campaigns and in some cases affects the pricing of impressions. The report also includes an NIZK verifying that the contents of the vectors of ciphertexts are an encoding of a acceptable report: we discuss the details of the NIZK in section 5.

The ad network's servers verify each of the reports by checking the NIZK, and sum together the vectors of ciphertexts submitted by users in each GeoIP bucket. They also implement any desired rate limiting to avoid click fraud. Ballots are only added together if they have the same demographic information, to permit statistics about demographic information to be used for campaigns. Furthermore, due to our optimizations described in §5.1, our system imposes a limit on the number of vectors that can be safely added together of around 2^{16}. So long as advertisements do not target extremely small numbers of users, this provides sufficient anonymity.

When the number of advertisements summed together begins to exceed this threshold, the ad server sends the totals to the trusted third party for decryption and resets the totals.

Given a summed ad ballot, the trusted party decrypts the aggregate ciphertext. This process consists of a single exponentiation per ciphertext every time the ad server transmits a vector. If the trusted third party is unreachable, the data may be stored for future decryption — thus we do not require the trusted third party to be as reliable as the overall system. Given the low complexity of the decryption operation, the trusted party can make use of a single core on a modest computing system to handle even 2 billion impressions per day. Note that although we assume a single trusted third party in this explanation, it is possible to employ threshold encryption and break the trusted party's role across multiple parties, each of which stores a share of the decryption key.

Following decryption, the ad server obtains g^s where s is the sum of the ballot entries at a given position. To recover s, the server must compute a discrete logarithm to recover the total number of advertisements. For small ad totals, brute force is sufficient — however, as we will discuss later in this paper, bandwidth optimizations suggest the use of precomputed tables as proposed by Bernstein and Lange in [13]. Through extensive use of pre-computation, the cost of this step can be kept small, although other tradeoffs between computation, table size, and client bandwidth are possible.

For additional privacy, this entire decoding process can be performed by the trusted third party who then further aggregates results before returning them. While this increases the load on the trusted third party, we still benefit from the 2^{16}-fold reduction in work via the summing performed by the network and the fact that the ad network has already handled proof validation. Our main scalability concerns are the size of the ciphertexts and zero knowledge proofs, the time it takes to verify the proofs, the time to compute the decryption, and the time to compute the discrete logarithm. In the next sections we discuss optimizations to the protocol that reduce these costs, and then discuss the measured performance we achieve on Amazon EC2 instances.

4. PRELIMINARIES

We review some of the standard constructions that will be used in our scheme.

Elgamal homomorphic encryption. Let $\mathbb{G}$ be a cyclic group of prime order p with generator g, and let $\Pi_{enc} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a variant of the Elgamal encryption scheme, defined as follows:

$\mathsf{KeyGen}(\mathbb{G})$. Select $sk \in_R \mathbb{Z}_p$ and output $pk = g^{sk}$, sk.

$\mathsf{Encrypt}(pk, m; r)$. For $m \in \{0, 1\}$, $r \in_R \mathbb{Z}_p$, output $C = (c_1, c_2) = (g^r, pk^r \cdot g^m)$.

$\mathsf{Decrypt}(pk, sk, C)$. Compute $h = c_2/c_1^{sk}$ and output $log_g(h)$.

[5] This is a second limitation of the Paillier-based implementation: because key sizes increase rapidly with number of advertisements, the correctness proofs only extend over a small number of advertisements, and do not limit the total number of impressions adequately.

Given a collection of ciphertexts $C_1, \ldots, C_N$ we denote $C^* = \prod_{i=1}^{N} C_i$ as $(\prod_{i=1}^{N} c_{i,1}, \prod_{i=1}^{N} c_{i,2})$. Upon successful decryption of C^* the output is equal to $\sum_{i=1}^{N} m_i$ where $m_1, \ldots, m_N$ represent the decryption of each ciphertext in the collection.

Pedersen commitments. Le $\mathbb{G}$ be a group of prime order p and let $g, h \in \mathbb{G}$ be two randomly-selected elements of $\mathbb{G}$ (such that no party knows the discrete logarithm of h w.r.t. g). Then to commit to a message $m \in \mathbb{Z}_p$, the committing party selects a random $r \in \mathbb{Z}_p$ and outputs $B = g^m h^r$. If the discrete logarithm problem is hard in $\mathbb{G}$, the Pedersen commitment scheme is computationally binding and unconditionally hiding.

An efficient proof that one commitment of a set opens to zero. Groth and Kohlweiss [25, §3] present an efficient Σ-protocol proof of knowledge that, for some set of (Pedersen) commitments $B_1, \ldots, B_N$, one of the commitments opens to 0. This proof has size of 4 $log\ N$ commitments and 3 $log\ N + 1$ elements of $\mathbb{Z}_p$. In the case where the protocol is being used to prove membership of a single commitment in a set there are several optimizations that reduce the work of a prover and verifier which we discuss later.

Private Information retrieval. Private information retrieval (PIR) allows a user to retrieve files from a server or set of servers without revealing which file is accessed. Computational PIR (cPIR), does so solely with cryptographic assumptions and without assuming non-cooperating parties. Critically, this means a cPIR scheme can be operated by one untrusted party and requires no trusted third parties.

5. AN EFFICIENT CRYPTOGRAPHIC PROTOCOL

The basic protocol used in Adnostic for ad reporting and our starting point is a simple voting scheme relying on a additively homomorphic encryption. A ballot for n "candidates" (in our case, each candidate represents an attribute related to an ad display) consists of a vector $\vec{c}$ of n additively homomorphic ciphertexts $c_1, c_2, \ldots, c_n$ encrypting bits $b_1 \ldots b_n$ and a proof π that all but one of those ciphertexts contains an encryption of 0. In other words, the proof ensures that the Prover has only "voted" once.

Because ciphertexts are additively homomorphic, the ad provider can compute the encrypted sum of the results $\vec{s} = \sum \vec{c_i}$ without viewing the contents of any individual ballot. They can then obtain the total number of impressions from a trusted third party who decrypts $\vec{s}$. While computing the sums for 2 billion impressions a day requires some computational effort, that cost is borne by the ad network: the trusted third party need only perform a single decryption.

The main bottleneck in this protocol is the proof of correctness π which the ad network must verify for each ballot. As discussed above, the approach taken by Adnostic and many voting schemes is to prove that each element c_i in $\vec{c}$ encrypts either 0 or 1, and then prove that $\sum c_i$ encrypts 1. This proof has $O(n)$ space complexity. We achieve a protocol with $O(log(n))$ space complexity and drastically improved performance by avoiding the individual proofs that each $c_i \in 0, 1$, while still enforcing the same property.

To do this, we select random values $a_0 \ldots a_n$, and for each ciphertext $c_i = g^{b+i} \cdot pk_i^{r_i}$, we compute $d_i = c_i^{a_i}$. Note, that if the ciphertexts are properly constructed, then the resulting $d = \sum d_i$ is a valid encryption of a single a_i as the remaining products evaluate to 0. Thus it remains simply to prove that the result is in fact an encryption of some a_i. To accomplish this, we leverage an efficient membership proof due to Groth and Kohlweiss [25].

Our initial construction outperforms even a heavily optimized and batched implementation of Elgamal with individual proofs per element. However, merely using the improved proof in the simple construction of Adnostic produces a protocol that is still too computationally expensive relative to the cost of each ad, and too bandwidth intensive to be practical for end-users. Obtaining the necessary performance requires making protocol-level modifications as well as several implementation-level optimizations. We now detail these.

5.1 Protocol Improvements

Reducing ciphertext size by using N public keys. The main limitation of the simple protocol is the cost of using a full Elgamal to encrypt a single bit. The naive Elgamal encryption requires two elements of $\mathbb{G}$ per bit encrypted, which (using elliptic curve groups with a 256-bit representation, at the 128-bit security level) implies a ciphertext expansion factor of approximately 512. This cost may prove too high for clients on low bandwidth connections.

To improve on the scheme, we first replace the single Elgamal public key pk with a vector of public keys $(pk_1, \ldots, pk_N)$. To encrypt a sequence of bits $b_1, \ldots, b_N$, the encryptor selects a random $r \in \mathbb{Z}_p$, outputs $(c_1, c_{1,2}, \ldots, c_{N,2}) = (g^r, pk_1^r \cdot g^{b_1}, \ldots, pk_N^r \cdot g^{b_N})$. Because the public keys are known a priori to both the client and the ad-network, they need not be sent and we halve our ciphertext expansion factor. This approach is well known to be secure under the assumption that the Decisional Diffie-Hellman problem is hard in $\mathbb{G}$ [12, 28].

Reducing ciphertext size by encoding multiple bits per ciphertext. To encode 4 bits per ciphertext, we compute $e_i = D^3 b_{4i+3} + D^2 b_{4i+2} + D b_{4i+1} + b_{4i+0}$ for some D, and then encrypt g^{e_i}. This modification means that at most D ciphertexts can be added together before decryption and decoding must happen, and the discrete log takes place in an interval of size D^4, thus increasing the cost of both of these steps. The benefit of this optimization is that it reduces the number of elements that must be processed in verifying the proof and reduces the bandwidth requirements of clients.

Include multiple impressions in the same ciphertext. We can include multiple impressions in a single ciphertext by modifying our proof statement to show that the sum is not 1, but rather sums to some number m. This modification adds a small cost cost linear in m. However, it enables a reduction in the amount of ciphertext bandwidth that each individual impression consumes by a factor m. Because this optimization is intricately interwound with the details our NIZK, we defer discussion of the details to the presentation of the final protocol.

5.2 Implementation Optimizations

Reducing computation using small exponents and multi-exponentiation. A separate approach to reducing the computational cost of proving and verification is to use short (*e.g.,* 128-bit) values for the elements $(a_1, \ldots, a_N)$. This reduces exponentiation cost, at the expense of an expected degree of security. The loss in security is proportional to the degree of shortening, as our proof demonstrates. This multi-exponentation is computed with Bos-Coster[22], and so the cost is proportional to the length of the elements.

Groth-Kohlweiss proofs require several multi exponentiations. We make use of the fact that we are proving membership of an exponent in a known set to replace one of the exponentiations with a polynomial interpolation, and use the Bos-Coster algorithm to compute the remaining multi-exponentiation. While Pippenger's algorithm [37] is also a possibility, we did not use it due to the complexity of implementing it, and the relatively low savings in the case of a massive number of random exponents.

Since we are encrypting a large number of elements in each impression we have adequate numbers of elements to get the full benefit of batching and so do not need to batch across separate impressions. Our Bos-Coster implementation uses a simple binary heap and reduces the number of limbs it iterates over whenever the maximum element no longer has that limb set.

Optimized Ed25519 implementations. We built our implementation on top of the Donna [34] implementation of the Ed25519 twisted Edwards curve. This implementation attains very high speed on 64-bit processors through careful choices of prime, addition law, and low-level multiplication strategies. We had to extend this implementation to include operations such as general scalar-point multiplication, which we did through a simple radix-16 add and multiply. We also carefully reordered calculations so as to expose opportunities for Strauss's algorithm and the use of precomputed tables in scalar multiplications. Our Bos-Coster implementation was inherited from this one, but required modifications to ensure that it would function with heaps that were not even in size and to enable its general use.

Precomputation for discrete logs We use the methods of Bernstein-Lange [13] to use a precomputed table to accelerate discrete log computations, taking a discrete log in an interval of size 2^{64} with 2^{12} additions. While not an optimization required for validating proofs, this step is necessary to ensure sufficiently low costs for the trusted third party.

Our discrete log computation uses batched inversion techniques to attain a high number of steps per second. The table is of size 2^{40}, and would cost around \$5,000 USD to compute on Amazon EC2. Alternatives that use smaller tables with more computations are certainly possible as discussed in Bernstein-Lange.

Uncompressed points. Transmitting compressed points requires the receiver to perform a square root calculation in order to obtain the decompressed point value. These square root calculations became a bottleneck, and so we used uncompressed points instead to eliminate them. This roughly doubles bandwidth, but as we will see the bandwidth requirements of our scheme are low due to earlier savings. It reduces the computation time of the ad server significantly.

5.3 Final protocol

We now discuss the NIZK for our optimized protocol. This proof uses ideas from the Groth-Kohlweiss [25] proof of membership as well as the classical Camenisch-Stadler [15] proofs of linear relations between logarithms.

We wish to encrypt a vector $b_0, b_1, \ldots b_N$ of bits, exactly one of which is 1. Our public key, whose private key is retained by the trusted third party, is $p_i = g^{k_i}$. We have the ciphertexts $c_i = g^{b_i} {p_i}^r$ which are transmitted along with the auxiliary value $K = g^r$. The trusted third party can decrypt these ciphertexts to obtain the value b_i.

Conceptually our proof that the c_i are well formed consists of taking a random vector $a_0, a_1, \ldots a_N$, computing $C = \prod_{i=0}^{N} c_i^{a_i}$ and $P = \prod^{N} p_i^{a_i}$, and demonstrating that (K, C) is a valid encryption of one of the a_i. This involves using proofs linear relations of discrete logarithms to relate C to a commitment to a single a_i, and then using a Groth-Kohlweiss proof to show that the commitment is indeed a commitment to one of the a_i.

As mentioned, our final protocol allows multiple bits to be encoded in a single ciphertext and allows multiple "votes" to be recorded in a single ballot. These modifications work as follows: fix D and m. m is a limit on the number of bits, while D will limit the number of ciphertexts that can be added together before overflow. D will be 2^{16} for our application. We will pack 4 bitfields into each element of the ciphertext.

Let $b_0, b_1, \ldots b_{4N}$ be the sequence of bits we want to encrypt, where m of them are 1 and the rest 0. Then define $e_i = \sum_{j=0}^{3} b_{4i+j} D^j$, and let $c_i = g^{e_i} p_i^r$. Then we transmit the c_i along with K. Our proof again takes a random vector $a_0, a_1, \ldots a_N$, and computes $C = \prod_{i=0}^{N} c_i^{a_i}$. The client also transmits m Pedersen commitments $E_1, \ldots E_m$ and $F_1, \ldots F_m$, and a proof of knowledge of an opening of each F_j to an $a_j \in a$, an opening of each E_j to $a_j \cdot (1 \vee D^1 \vee D^2 \vee \ldots d^M)$ and that (K, C) is an encryption of the same value as the sum of the E_i. This proof is again a combination of a Groth-Kohlweiss proof and a Camenisch-Stadler style proof of a set of equations.

As described the proof is interactive. We show honest verifier zero-knowledge and soundness of this interactive proof, then apply the Fiat-Shamir transform to obtain an non-interactive proof. The standard zero-knowledge proof that the encryption satisfies certain properties is thus a black box in our proof.

We fix $g, h \in \mathbb{G}$ a group of prime order where the decisional Diffie-Hellman problem is hard, where h is an element with unknown discrete logarithm with respect to g. Our protocol starts with the transmission of $K, c_0, \ldots, c_N$. The verifier responds with a random sequence $a_0, \ldots a_N$, and both compute $C = \prod_i c_i^{a_i}$. The prover now transmits $E_1, E_2, \ldots E_m$, $F_1, \ldots F_m$ and a zero-knowledge proof of knowledge for the following statement: *There exist r, s, z_i, v_i, y_i and t_i such that*

$$K = g^r \tag{1}$$
$$C = g^s P^r \tag{2}$$
$$s = \sum z_i \tag{3}$$
$$E_i = g^{z_i} h^{v_i} \tag{4}$$
$$F_i = g^{y_i} h^{t_i} \tag{5}$$
$$z_i = y_i \vee z_i = D y_i \vee z_i = D^2 y_i \vee z_i = D^3 y_i \tag{6}$$
$$y_i \in \{a_i\} \tag{7}$$

Parts 1 - 6 are proven via standard techniques and 7 is performed using the membership roof of Groth-Kohlweiss [25].

Security analysis. We now demonstrate soundness. We can always write $c_i = p_i^r g_i^e$, and then we have $s = \sum_{i=0}^{N} a_i e_i$. The proved statement implies that $s = \sum_{j=0}^{m} D^{b_j} a_{w_j}$ for some b_i in $0, 1, 2, 3$, and w_i in $1, 2, \ldots, N$. There are $(4N)^m$

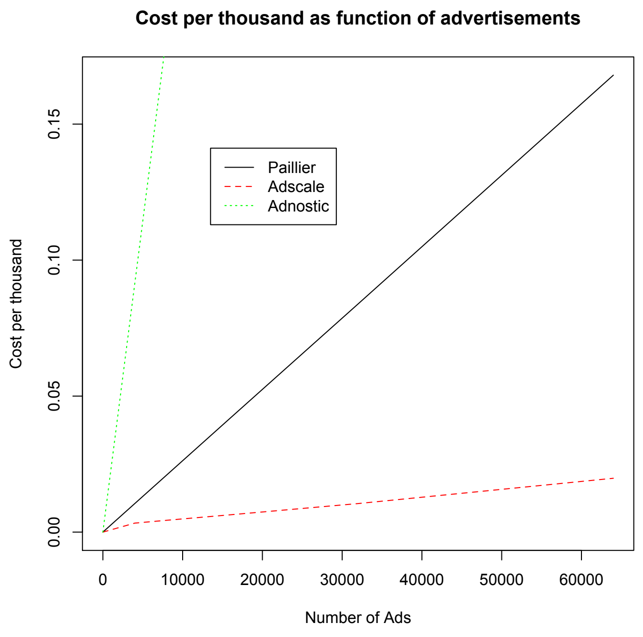


Figure 2: Performance of our scheme vs. others

such choices. We now have that $\sum a_i(e_i - q_i) = 0$ where q_i is a correctly formed vector of exponents. Suppose e_i was not correctly formed, then each of the $(4N)^m$ such vectors is nonzero. But the probability of finding a q_i that will satisfy the equation has an upper bound of $(4N)^m/S$ if there are S possibilities for each a_i by the Schwartz lemma [40]. Therefore with probability $1-(4N)^m/S$ cheating is detected. S will be 2^{128}, and for $N = 2^{16}$ and $m = 3$, the probability of successful cheating is at most 2^{-80}.

To prove honest-verifier zero-knowledge we note that the commitments $E_1, E_2, \ldots E_m$, and $F_1, F_2 \ldots F_n$ are statistically indistinguishable from random group elements. Our simulator picks random values for these, then executes the simulator for the zero knowledge proof used later on. Therefore the distribution of transcripts is independent of which correctly formed ciphertext was sent, and thus reveals no information about what was encrypted.

6. EXPERIMENTS

6.1 Ad Reporting via Homomorphic Encryption

To validate the performance of our scheme we compare it to two other schemes. The first scheme encrypts each bit of the vector with Elgamal and includes a separate proof of correctness for each bit, along with a proof that the sum of the ciphertexts is an encryption of 1. These proofs are designed to be efficiently batch validated, following the general model of Camenisch and Stadler [15]. The second scheme uses a Paillier based approach due to Groth [24]. Unfortunately this scheme is less secure then the alternatives, as it does not prove that the number of ads selected is limited, but only that at most one out of a hundred ads was selected. These alternative schemes were extensively optimized before this new scheme was devised after it became clear they could not meet our demands.

Our measurements are in cost per mille as a function of the number of advertisements with current EC2 pricing. We show the graph of our results in figure 2. We measured the time taken for verification over 100-1000 impressions at each given number of advertisements. This measurement does not include the time taken by the trusted third party, or the time to add the vectors together, or the time for discrete logarithm computations in any scheme. Each impression produces some amount of revenue, and therefore scaling linearly with impressions is reasonable for any scheme. However, each additional advertisement does not necessarily produce additional revenue, and therefore the cost per additional advertisement should be as low as possible.

The trusted third party needs to only compute a decryption of an Elgamal ciphertext, and furthermore handles $1/2^{16}$ of the data that the verifier does as the ciphertexts have been summed before decryption. A maximum of 2^{16} ciphertexts can be summed before counter overflow threatens the results, but this provides adequate security against deanonymization. The trusted third party has an extremely small cost even when handling 2 billion impressions a day. The discrete log calculation is more intensive, but as our precomputed table enables a single core to compute 153 discrete logs per second, and the discrete log can be applied only to $1/2^{16}$ of the data, the cost is only one percent of the cost of verification in our scheme, even for large numbers of advertisements and impressions.

The bandwidth and verification time consumption are presented in Figure 3. Because of our use of logarithmic size proofs, the scaling is slightly sublinear, but quickly approaches linear scaling in the number of advertisements. As shown in Table 1 (at the start of this paper) these costs are significantly lower than for the alternatives. This is because we are able to combine multiple impressions into a single ballot, as well as packing multiple bits into a single ciphertext, and our proof has logarithmic size in addition to the ciphertexts, rather then linear. Even on constrained connections and with large numbers of advertisements our bandwidth consumption per impression is reasonable.

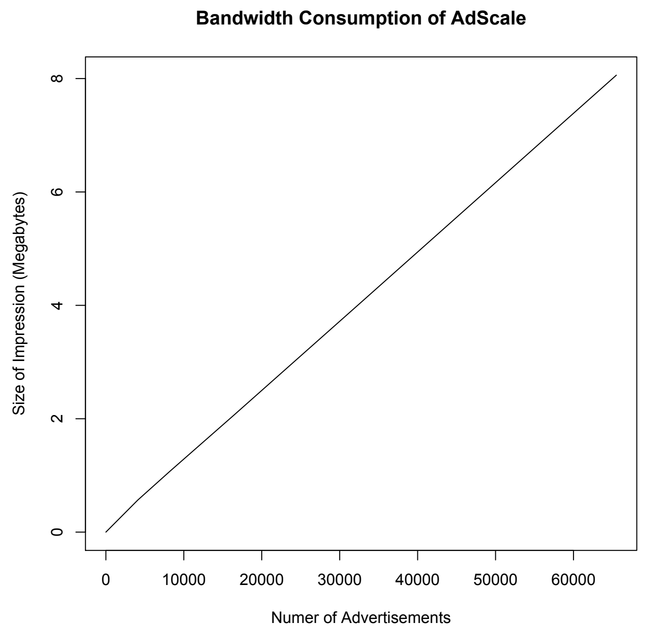
We conducted all our measurements on Amazon Web Services, C4.xlarge instances, and used current per-hour pricing to determine the cost figures. While computation costs vary with provider and technology, as does the performance of the underlying system, the ratios between our scheme's performance and the alternatives will remain largely the same.

6.2 Ad Retrieval via cPIR

As a complement to our ad reporting protocol, we additional measured a related aspect: the performance of ad delivery via computational information retrieval. We assume ads are 40KB each. We use the cPIR scheme implemented in [8]. The implementation consists of a server which serves files from a directory—in our case a directory of ads—and a client to retrieve them.

The XPIR client does an optimization pass to determine the ideal parameters to use given available bandwidth and processing power. The optimizer can minimize resources spent, round trip time, or cost to operate in the cloud. We modify the optimization pass to minimize bandwidth.

Our costs are computed on an Amazon EC2 C4.2xlarge with an Intel Xeon E5-2666 v3 @ 2.9GHz and 32GB of ram,



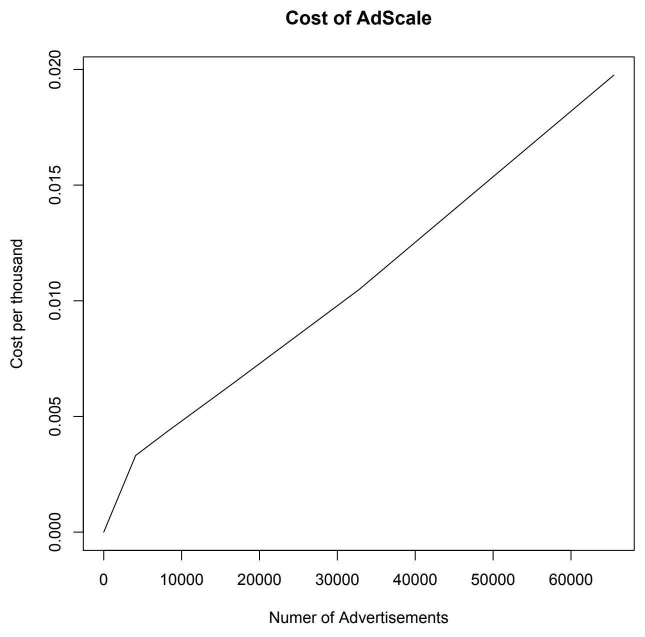


Figure 3: Performance of our scheme alone

Ads	time (seconds)	CPM	up (MB)	down(MB)
10,000	0.1504 ± 0.003	0.002	3.2	2.3
50,000	0.73 ± 0.020	0.008	2.62	7.34
100,000	1.43 ± 0.04	0.02	10.37	2.62

(a) cPIR performance

Total files	size of files	ads per file	expansion factor
100	4mb	100	7x
1,000	.4mb	10	47x
10,000	40kb	1	137x

(b) cPIR bandwidth expansion for constant size db

giving us a cost of 4 cents per core per hour. The files are stored in memory. Bandwidth is measured directly by instrumenting the client. We run 10 iterations.

As can be seen in table 2a, the costs for cPIR are surprisingly reasonable. We note there is a marked decrease in efficiency between 50k and 100k ads. This appears to be due to the optimizer selecting different parameters at that scale, not a performance cliff, and that other trade offs are possible.

The bandwidth costs of the cPIR scheme are relatively high. PIR schemes in general simply do not operate very efficiently over many small files. Combining multiple advertisements into one file may enable a single download to cover multiple impressions, and is already an optimization used in the cPIR scheme profiled here. However, selecting advertisements so that this happens is likely to require some careful consideration, perhaps by putting ads together by category. We do not further consider this question here due to the uncertainties about how effective it will be.

CPIR Feasibility. Assuming no batching, for the client to retrieve 20 ads per day at $2.62 + 7,34 \approx 10$ MB per ad would require the client to download 200 MB per day. This would take approximately 200 seconds on a 1 MB/s Internet connection. For reference, according to the Federal Communication Commission's National Broadband Map, in 2014 the median household Internet connection download bandwidth was 6.7 MB/s, with 80% of home users having access to greater than 2.4 MB/s connections (res. 1MP/s and 0.5MP/s for upload) [2]. As ad data can be downloaded during idle periods through the preceding day, this is surprisingly feasible for a home computer which is always connected to the Internet.

The two larger limitations seem to be overall network capacity and the effect on users with bandwidth caps. In terms of network impact, we note that Netflix streaming takes at least 2 MB/s a second [30] and that this usage is concentrated in the evenings. In contrast, downloading ads can be distributed over idle time. The larger limitation is that at 6GB a month, users might run afoul of bandwidth caps (e.g. ad retrieval would constitute 4% of AT&T's monthly 150GB cap.[14]).

After 10 days the trivial solution of downloading all advertisements becomes more efficient in bandwidth terms. Unfortunately for extremely large numbers of advertisements, particularly images, disk consumption becomes an issue.

7. CONCLUSION

We have demonstrated a system with significantly better bandwidth and security properties then existing approaches to privacy preserving advertising. While advertisement distribution has several complexities still to be worked out, we have solved the reporting problem without requiring trusted third parties to scale, without unacceptable bandwidth consumption, and without prohibitive computational costs. Our scheme has overhead costs of less then 1 cent per thousand impressions, even when tens of thousands of advertisements are being displayed, requires only a single core for a third party to use, and uses only 1 megabyte of bandwidth per impression when handing $32,000$ advertisements, properties that existing work in the area does not have.

Acknowledgements

This research was supported in part by the National Science Foundation under awards CNS-1010928 and EFMA-1441209; The Office of Naval Research under contract N00014-14-1-0333; and the Mozilla Foundation.

8. REFERENCES

[1] Brave web browser. https://www.brave.com/.

[2] National Broadband Map. http://www.broadbandmap.gov/summarize/nationwide, June 2014.

[3] Tor metrics — bridge users by country. https://metrics.torproject.org/userstats-bridge-country.html?start=2015-2-14&end=2016-02-14&country=all, 2016. Online; accessed 14 February 2016.

[4] Tor metrics — direct users by country. https://metrics.torproject.org/userstats-relay-country.html?start=2015-02-14&end=2016-02-14&country=all&events=off, 2016. Online; accessed 14 February 2016.

[5] Tor metrics — total relay bandwidth in the network. https://metrics.torproject.org/bandwidth.html, 2016. Online; accessed 14 February 2016.

[6] Tor metrics bridge users by country. https://metrics.torproject.org/userstats-relay-country.html, 2016. Online; accessed 21 May 2016.

[7] Adobe. The cost of ad blocking. Available at https://downloads.pagefair.com/wp-content/uploads/2016/05/2015_report-the_cost_of_ad_blocking.pdf, 2015.

[8] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. Xpir: Private information retrieval for everyone. Cryptology ePrint Archive, Report 2014/1025, 2014. http://eprint.iacr.org/.

[9] Tiago Alves and Don Felton. Trustzone: Integrated hardware and software security. *ARM white paper*, 3(4):18–24, 2004.

[10] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative technology for cpu based attestation and sealing. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, volume 13, 2013.

[11] Michael Backes, Aniket Kate, Matteo Maffei, and Kim Pecina. Obliviad: Provably secure and practical online behavioral advertising. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 257–271. IEEE, 2012.

[12] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT '00*, pages 259–274. Springer, 2000.

[13] Daniel J. Bernstein and Tanja Lange. Computing small discrete logarithms faster. In Steven D. Galbraith and Mridul Nandi, editors, *Progress In Cryptology-INDOCRYPT 2012*, number 7668 in Lecture Notes in Computer Science, pages 317–338. Springer, December 2012.

[14] Jon Brodkin. AT&T and Verizon defend data caps on home Internet service. http://arstechnica.com/business/2014/09/att-and-verizon-defend-data-caps-on-home-internet-service/, September 2014.

[15] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical Report 260, Department of Computer Science, ETH Zurich, 1997.

[16] Farah Chanchary and Sonia Chiasson. User perceptions of sharing, advertising, and tracking. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 53–67, Ottawa, July 2015. USENIX Association.

[17] Henry Corrigan-Gibbs and Bryan Ford. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 340–350. ACM, 2010.

[18] Henry Corrigan-Gibbs, David Isaac Wolinsky, and Bryan Ford. Proactively accountable anonymous messaging in verdict. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 147–162, Washington, D.C., 2013. USENIX.

[19] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European transactions on Telecommunications*, 8(5):481–490, 1997.

[20] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, SP '03, Washington, DC, USA, 2003. IEEE Computer Society.

[21] George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 269–282. IEEE, 2009.

[22] Peter de Rooij. *Advances in Cryptology — EUROCRYPT'94: Workshop on the Theory and Application of Cryptographic Techniques Perugia, Italy, May 9–12, 1994 Proceedings*, chapter Efficient exponentiation using precomputation and vector addition chains, pages 389–399. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.

[23] Rafi Goldberg. Lack of trust in internet privacy and security may deter economic and other online activities privacy and security may deter economic and other online activities. Available at https://www.ntia.doc.gov/blog/2016/lack-trust-internet-privacy-and-security-may-deter-economic-and-other-online-activities, May 2013.

[24] Jens Groth. Non-interactive zero-knowledge arguments for voting. In *Theory of Cryptography*, number 3378 in Lecture Notes in Computer Science, 2005.

[25] Jens Groth and Markulf Kohlweiss. One-out-of-many

proofs: Or how to leak a secret and spend a coin. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 253–280. Springer Berlin Heidelberg, 2015.

[26] Ryan Henry, Femi Olumofin, and Ian Goldberg. Practical pir for electronic commerce. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 677–690, New York, NY, USA, 2011. ACM.

[27] Ari Juels. Targeted advertising... and privacy too. In *Topics in Cryptology—CT-RSA 2001*, pages 408–424. Springer, 2001.

[28] Myungsun Kim, Jihye Kim, and Jung Hee Cheon. Compress multiple ciphertexts using elgamal encryption. Cryptology ePrint Archive, Report 2012/243, 2012. http://eprint.iacr.org/2012/243.

[29] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. A case study on measuring statistical data in the Tor anonymity network. In *Proceedings of the Workshop on Ethics in Computer Security Research (WECSR 2010)*, LNCS. Springer, January 2010.

[30] Jim Martin, Yunhui Fu, Nicholas Wourms, and Terry Shaw. Characterizing netflix bandwidth consumption. In *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, pages 230–235. IEEE, 2013.

[31] Jon McLachlan, Andrew Tran, Nicholas Hopper, and Yongdae Kim. Scalable onion routing with torsk. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 590–599. ACM, 2009.

[32] Silvio Micali and Ronald L. Rivest. Micropayments revisited. In *Proceedings of the Cryptographer's Track at the RSA Conference*, CT-RSA '02, pages 149–163, 2002.

[33] Prateek Mittal, Femi G Olumofin, Carmela Troncoso, Nikita Borisov, and Ian Goldberg. Pir-tor: Scalable anonymous communication using private information retrieval. In *USENIX Security Symposium*, 2011.

[34] Andrew Moon. Ed25519-donna. https://github.com/floodyberry/ed25519-donna, 2014.

[35] Arjun Nambiar and Matthew Wright. Salsa: a structured approach to large-scale anonymity. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 17–26. ACM, 2006.

[36] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology—EUROCRYPT'99*, pages 223–238. Springer, 1999.

[37] Nicholas Pippenger. On the evaluation of powers and related problems (preliminary version). In *17th Annual Symposium on Foundations of Computer Science, Houston, Texas, USA, 25-27 October 1976*, pages 258–263. IEEE Computer Society, 1976.

[38] PricewaterhouseCoopers. Iab internet advertising revenue report. http://www.iab.com/wp-content/uploads/2015/05/IAB_Internet_Advertising_Revenue_FY_2014.pdf.

[39] Ronald L. Rivest. Peppercoin micropayments. In *Proceedings of the 8th International Conference on Financial Cryptography*, FC '04, pages 2–8, 2004.

[40] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980.

[41] Robin Snader and Nikita Borisov. A tune-up for tor: Improving security and performance in the tor network. In *NDSS*, volume 8, page 127, 2008.

[42] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. Adnostic: Privacy preserving targeted advertising. In *Proceedings Network and Distributed System Symposium*, 2010.

[43] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 179–182, 2012.

APPENDIX

A. LIMITATIONS OF TOR

Some recent proposals [1] have considered handling both ad distribution and reporting over Tor. In this section we discuss the viability of using the Tor network for this purpose.

Tor handles 75Gb/s [5] per second today as measured by methods discussed in [29], and the bandwidth consumed by reporting advertisement impressions will be a small fraction of that. Indeed, even distributing ads is feasible needing only 7.4Gb/s. However, the number of connections required is a significant barrier.

Between February 14 2015 and February 14, 2016, Tor averaged no more than 2.3 Million[4] direct users and an average of no more than 40,000 bridge users with a highly abnormal peak of 140,000 in June[3]. Operating a privacy preserving ad network at scale over Tor would thus result in a $4,000\%$ increase in the number of users . Far more significantly, it would result in a 40–fold to 80–fold increase in the number of circuits. In order for ad impressions to be unlinkable even from each other, each of the 20 impressions per user per day must be submitted using a separate circuit. In contrast, normal user circuits are relatively long lived. It is unlikely Tor can scale to these levels without substantial re-engineering e.g. as proposed in [33].

Furthermore, Tor is frequently blocked on corporate networks and in entire countries. This would result in ad impressions being never reported, and thus lost revenue. The latency incurred by downloads through Tor would make immediate responses to search queries in the form of advertisements impossible, and this is a particularly valuable source of information about what users are looking for. Reporting ad impressions through Tor also makes rate-limiting of malicious users difficult: solutions based on anonymous credentials are possible, but the issuance of these credentials and reissuance in case of loss poses challenges.