Combating Click Fraud via Premium Clicks

Ari Juels
RSA Laboratories[1]
ajuels@rsa.com

Sid Stamm
Indiana University, Bloomington
sstamm@indiana.edu

Markus Jakobsson
Indiana University, Bloomington and RavenWhite Inc.
markus@indiana.edu

Abstract

We propose new techniques to combat the problem of click fraud in pay-per-click (PPC) systems. Rather than adopting the common approach of filtering out seemingly fraudulent clicks, we consider instead an affirmative approach that only accepts legitimate clicks, namely those validated through client authentication. Our system supports a new advertising model in which "premium" validated clicks assume higher value than ordinary clicks of more uncertain authenticity. Click validation in our system relies upon sites sharing evidence of the legitimacy of users (distinguishing them from bots, scripts, or fraudsters). As cross-site user tracking raises privacy concerns among many users, we propose ways to make the process of authentication anonymous. Our premium-click scheme is transparent to users. It requires no client-side changes and imposes minimal overhead on participating Web sites.

Key words: authentication, click-fraud

1 Introduction

Pay-per-click (PPC) metering is a popular payment model for advertising on the Internet. The model involves an *advertiser* who contracts with a specialized entity, which we refer to as a *syndicator*, to distribute textual or graphical banner advertisements to *publishers* of content. These banner ads point to the advertiser's Web site: When a user clicks on the banner ad on the publisher's webpage, she is directed to the site to which it points. Search engines such as Google and Yahoo are the most popular syndicators, and create the largest portion of pay-per-click traffic on the Internet today. These sites display advertisements on their own search pages in response to the search terms entered by users and charge advertisers for clicks on these links (thereby acting as their own publishers) or, increasingly, outsource advertisements to third-party publishers. Advertisers pay syndicators per referral, and the syndicators pass on a portion of the payments to the publishers.

A syndicator or publisher's server observes a "click" simply as a browser request for a URL associated with a particular ad. The server has no way to determine if a human initiated the action—and, if a human was involved, whether she acted knowingly and with honest intent. Syndicators typically seek to filter fraudulent or spurious clicks based on information such as the type of advertisement that was requested, the cost of the associated keyword, the IP address of the request and the recent number of requests from this address. In this paper, we propose an alternative approach. Rather than seeking to detect and eliminate fraudulent clicks, i.e., filtering out seemingly bad clicks, we consider ways of *authenticating valid clicks*, i.e., admitting only verifiably good ones. We refer to such validated clicks as *premium clicks*.

Our scheme involves a new entity, referred to as an *attestor*, that provides cryptographic credentials for clients that perform qualifying actions, such as purchases. These credentials allow the syndicator to distinguish premium clicks–corresponding to relatively low-risk clients–from other, general click traffic. Such classification of clicks strengthens a syndicator's heuristic isolation of fraud risks.

The premium-click techniques that we describe in this paper are complementary to existing, filter-based tools for validating clicks: The two approaches can can operate side by side.

Organization. We begin with a problem statement and a description of the the related work in section 2, followed by a structural overview of our approach in section 3. In section 4, we outline our scheme and detail its technical foundations. We describe a prototype implementation of our scheme in section 5 and discuss user privacy in section 6, proposing several privacy-enhancing techniques. We provide a brief security analysis in section 7, and conclude in section 8. The paper appendix describes design choices for premium-click systems with multiple attestors.

2 Problem Overview and Related Work

Click-fraud is a type of abuse that exploits the lack of verifiable human engagement in PPC requests in order to fabricate ad traffic. It can take a number of forms. One virulent, automated type of click fraud involves a client that fraudulently simulates a click by means of a script or bot—or as the result of infection by a virus or Trojan. Such malware typically resides on the computer of the user from which the click will be generated, but can also in principle reside on access points and consumer routers [8, 9, 7]. Some click-fraud relies on *real* clicks, whether intentional or not. An example of the former is a so-called click-farm, which is a term denoting a group of low-wage workers who click for a living; another example involves deceiving or convincing users to click on advertisements. An example of an *unintentional* click is one generated by a malicious cursor-following script that places the banner right under the mouse cursor [6]. This can be done in a very small window to avoid detection. When the user clicks, the click would be interpreted as a click on the banner, and cause revenue generation to the attacker. A related abuse is manifested in an attack where publishers manipulate web pages such that honest visitors inadvertently trigger clicks [4]. This can be done for many common PPC schemes, and simply relies on the inclusion of a JavaScript component on the publisher's webpage, where the script reads the banner and performs a get request that corresponds to what would be performed if a user had initiated a click.

Click fraud can benefit a fraudster in at least three known ways: First of all, a fraudster can use click-fraud to inflate the revenue of a publisher. Second, a fraudster can employ click-fraud to inflate advertising costs for a commercial competitor. As advertisers generally specify caps on their daily advertising expenses, such fraud is essentially a denial-of-service attack. Third, a fraudster can modify the ranking of advertisements by a combination of impressions and clicks. An impression is the viewing of the banner, with no click; this causes the ranking of the associated advertisement to go down. This can be done to benefit own advertising programs at the cost of those of competitors, and to manipulate the price paid per click for selected keywords.

Syndicators can in principle derive financial benefit from click fraud in the short term, as they receive revenue for whatever clicks they deem "valid." In the long term, however, as customers become sensitive to losses, and syndicators rely on third-party auditors to lend credibility to their operations, click fraud can jeopardize syndicator-adverstiser relationships. Thus syndicators ultimately have a strong incentive to eliminate fraudulent clicks. Today they employ a battery of filters to weed out suspicious clicks. These filters are trade secrets, as their disclosure might prompt new forms of fraud [10]. To give one example, though, it is likely that syndicators use IP tracing to determine if an implausible number of clicks is originating from a single source. While heuristic filters are fairly effective, they are of limited utility against sophisticated fraudsters, and subject to degraded performance as fraudsters learn to defeat them.

3 Structural Overview

Authentication. Our premium-click scheme is based on authentication of requests via cryptographic attestations on client behavior. We refer to these attestations as *coupons*. While a coupon could be realized straightforwardly using traditional *third-party cookies*, such cookies are so commonly blocked by consumers that their use is often impractical. Our scheme could alternatively involve traditional first-party cookies dispensed and harvested by a central authority. This architectural approach, however, presents limitations that we explain in depth in section 4.1. As we explain, we instead focus in this paper on the alternative mechanism of *cache cookies*.

Our premium-click scheme has two distinctive aspects:

1. **Pedigree:** Our scheme relies on designated Web sites called *attestors* to identify and label clients that appear to be operated by legitimate users—as opposed to bots or fraudsters. For example, an attestor might be a retail Web site that classifies as legitimate any client that has made at least $50 of purchases. (Financial commitment here corroborates legitimate user behavior.) We refer to such clients, the producers of premium clicks in our scheme, as *premium clients*. In a loose sense, we propose the creation of an implicit reputation network to combat click-fraud, much like the seller reputation on eBay [3].

2. **Traffic caps:** Our scheme supports validation of clicks from clients that have not produced an excessive degree of click-traffic and thereby indicated possible malicious activity. In our approach, a click is only regarded as valid if accompanied by a coupon. Thus, we can detect multiple requests from the same origin by keeping track of coupon presentation. In the standard approach in which attestations like coupons do not play a role, detection of same-source traffic is more challenging, and often depends upon coarser origination data, such as IP addresses, or more fragile markers of continuity, such as session identifiers.

Architecture. In a traditional scheme, as a user clicks on a banner placed on the site of a publisher, the corresponding advertisement is downloaded from the advertiser and the transaction recorded by the syndicator. Later, the syndicator bills the advertiser and pays the publisher.

Under the model of premium clicks, there are additional tasks carried out: As a user performs a qualified action (such as a purchase), the corresponding attestation is embedded in his browser by an attestor. This attestation is released to the syndicator when the user clicks on a banner. The release can be initiated either by the syndicator or the advertiser. (Our prototype relies on syndicator triggering of coupon release.) The syndicator can pay attestors for their participation in a number of ways, ranging from a flat fee per time period to a payment that depends on the number of associated attestations that were recorded in a time interval. To avoid a situation where dishonest attestors issue larger number of attestations than the protocol prescribes (which would increase the earnings of the dishonest attestors), it is possible to appeal to standard auditing techniques.

Challenges. Our approach to premium clicks gives rise to two technical challenges. First, we must securely validate premium clients and their associated clicks. In other words, we must ensure that adversaries cannot impersonate premium clients or forge premium clicks. For this purpose, we apply basic cryptographic tools for data integrity. Second, we must protect the privacy of clients. While we do want syndicators to be able to authenticate clients, we do not want syndicators to be able to track them, learn their identities, or harvest side information about their browsing patterns. Toward this end, we propose ways in which coupons may be created as essentially anonymous credentials.

Of course, our techniques do not prevent misuse of coupons by clients that are "good," i.e., controlled by honest users, and then turn "bad," e.g., become infected with malware. By identifying the sources of clicks, however, and making traffic caps more effective, coupons in our scheme still offer some protection against fraud even in such cases.

It is important to observe that existing filtering methods cannot in general employ cookies/coupons to detect fraudulent clicks. That is because filtering is an *exclusionary* process: It seeks to identify and eliminate "bad" clicks. If a cookie were used to mark and exclude certain types of "bad" users, fraudsters could simply remove the cookies from their browsers. In contrast, because our premium-click scheme is *distinguishing*, i.e., it only accepts "good" clicks, it can benefit from the use of cookies/coupons. Cookies serve to mark "good" users.

4 A Premium-Click Scheme

In a world of perfect transparency, in which a syndicator knew the (real-world) identity of all users clicking on ads, click fraud would be much more manageable. In such a world, it would be easier to identify misbehavior by a real user—e.g., implausibly many clicks—as well as clicks initiated by bogus users or bots. A syndicator could go further, and reference databases containing profiles on the users who clicked on its published ads. The syndicator could even create a highly refined pricing structure based on a user's predicted value as a potential consumer, with differential compensation for publishers. Our premium-click protocol diverges from this ideal in two senses:

- **Partial knowledge:** Given the fragmented nature of databases on user behavior and the privacy concerns attendant on user profiling, our overall profiling goal is modest. We would like to enable a syndicator only to determine that a click originates with a true human user with probable honest intent. We do not mainly focus on stronger differentiation among users, although our protocols could support this goal.

- **The browser as carrier:** Rather than relying on a central data repository, we rely on users' browsers to convey information among participating sites. This approach helps eliminate engineering complexity and protect user privacy.

We design our premium-click scheme to support outsourced PPC advertising. It can equally well secure against click fraud when ads are published directly on search engines: We need simply treat the syndicator and publisher as the same entity. The steps in our scheme are as follows and are illustrated in Figure 1. For simplicity, we assume a single syndicator $\mathcal{S}$ and attestor $\mathcal{A}$. (We discuss the case of multiple attestors in the appendix.)

1. **Marking:** Based on its criteria for user validation, the attestor identifies a visiting client as legitimate. The attestor then "marks" the client. It does so by caching in the client's browser a *coupon* γ, a kind of cryptographic token.

2. **Click / coupon release:** When a user clicks on a publisher's advertisement in a browser, the user's browser is directed to a URL on the syndicator's site. This URL includes the publisher's identity ID_{pub} and the identity of the advertisement that was clicked with ID_{ad}. The syndicator then causes the browser to release its coupon γ simultaneously with ID_{pub} and ID_{ad}.[2] We let $C = (\gamma, ID_{pub}, ID_{ad})$

denote the released triple. We shall henceforth refer to γ or C alternately as a "coupon," according to context.

3. **Coupon checking:** On receiving a triple $C = (\gamma, ID_{pub}, ID_{ad})$, the syndicator checks that γ is a (cryptographically) well formed coupon, as we describe in depth later. The syndicator also checks that the coupon has not been over-used, i.e., that C has not been submitted an excessive number of times in the recent past. (What constitutes "excessive" submission is a policy decision.)

4. **Reward:** If the syndicator successfully verifies that C represents a valid premium click, then the syndicator pays the publisher accordingly.

Of course, the publisher might embed additional information in C, e.g., a timestamp, etc. Moreover, a user's browser might in fact contain multiple coupons $\gamma_1, \gamma_2, \ldots$ from different attestors, a possibility that we discuss below. A single computer may have multiple users, of course. If they each maintain a separate account, then their individual browser instantiations will carry user-specific coupons. When users share a browser, the browser may carry coupons if at least one of the users is validated by an attestor. While validation is not user-specific in this case, it is still helpful: A shared machine with a valid user is considerably more likely to see honest use than one without.

We now detail the technical foundations of our scheme. We assume here that the browser of a given user carries at most one coupon. We address the case of multiple coupons in the appendix.

4.1 Coupon caching

Our first technical design choice is the transport medium for coupons. To ensure its correct association with the browser that created it, a coupon is best communicated as a cached browser value (rather than through a back channel). At the same time, it is important to ensure that coupons be set such that only the syndicator can retrieve them, and fraudsters cannot easily harvest them.

Third-party cookies are the most obvious way to instantiate coupons. A third-party cookie is one set for a domain other than the one being visited by the user; thus, a coupon could be set as a third-party cookie. Because third-party cookies have a history of abusive application, however, users regularly block them. First-party cookies are an alternative mechanism. If an attestor redirects users to the site of a syndicator and provides user-specific or session-specific information in the redirection, then the syndicator can implant a coupon in the form of a first-party cookie for its own, later use. Redirection of this kind, however, can be cumbersome, particularly if an attestor has relationships with multiple syndicators.

Cache cookies [5], particularly the TIF-based variety, offer an attractive alternative. An attestor can embed a coupon in a cache-cookie that is tagged for the site of a syndicator, i.e., exclusively readable by the syndicator. In their ability to be set for third-party sites, cache cookies are similar in functionality to third-party cookies. Cache cookies have a special, useful quirk, though: *Any* Web site visited by a user can cause them to be released to the site for which they are tagged. (Thus, as we shall see, it is important to authenticate the site initiating their release from a user's browser.) Cache cookies, moreover, function even in browsers where ordinary cookies have been blocked. Cache cookies are therefore our preferred medium for coupons.

Briefly, a TIF-based cache cookie works as follows. Suppose we wish to set a cache cookie bearing value γ for release to Web site www.S.com. The cache cookie, then, assumes the form of an HTML page ABC.html that requests a resource from www.S.com bearing the value γ. For example, ABC.html might display a GIF image of the form http://www.S.com/γ.gif. Observe that *any* Web site can create ABC.html and plant it in a visiting user's browser. Similarly *any* Web site that knows the name of the page/cache-cookie ABC.html can reference it, causing www.S.com to receive a request for γ.gif. Only www.S.com, however, can receive the cache cookie, i.e., the value γ, when it is released from the browser.

4.2 Coupon authentication

Ensuring against fraudulent creation or use of coupons is a key challenge in our scheme. Only attestors should be able to construct valid coupons. Coupons must therefore carry a form of cryptographic authentication. While digital signatures can in principle offer a flexible way to authenticate coupons, their computational costs are probably prohibitively expensive for a high-traffic, potentially multi-site scheme of the type we propose here. Message-authentication codes (MACs), a symmetric-key analog of digital signatures, are a more practical alternative.[3]

Suppose that the attestor $\mathcal{A}$ and syndicator $\mathcal{S}$ share a symmetric key k. (This key may be established out of band or using existing secure channels.) Let $MAC_k(m)$ represent a strong message authentication code, e.g., HMAC [2], computed on a suitably formed message m. It is infeasible for any third party, e.g., an adversary, to generate a fresh MAC on any message m. Consequently, if a coupon assumes the form $\gamma = m \parallel MAC_k(m)$ for a bitstring m that is unique to the visit of a client to the site of an attestor, then the coupon can be copied, but cannot

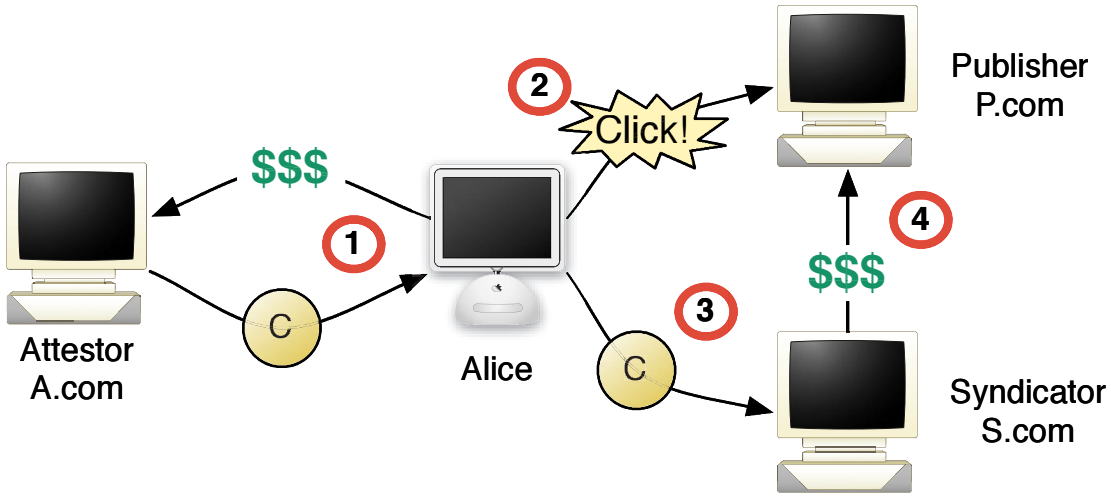

Figure 1: (1) **Alice** visits attestor **A.com**, spends money, and receivs coupon value $C = \gamma$. (2) **Alice** visits publisher **P.com** and clicks on an ad. (3) **Alice's** browser transmits coupon $C = (\gamma, ID_{pub}, ID_{ad})$ to syndicator **S.com**. (4) **S.com** pays **P.com** for Alice's premium click. (S.com also redirects Alice's browser to the entity that created the ad.)

be feasibly modified by a third party. The value m might be a suitably long (say, 128-bit) random nonce generated by $\mathcal{A}$. We propose some privacy-protecting alternative formats for m below.

4.3 Publisher identification/authentication

In addition to ensuring that a coupon is authentic, a syndicator must also be able to determine what publisher caused it to be released and is to receive payment for the associated click. Recall from above that a coupon takes the form $C = (\gamma, ID_{pub}, ID_{ad})$, where ID_{pub} is the identity of the publisher and ID_{ad} identifies the advertisement clicked. In order to create a full coupon, we must append ID_{pub} and ID_{ad} to γ as it is released. To do so, we can enhance a cache cookie webpage X.html to include the document *referrer*, i.e., the tag that identifies the webpage that causes its release. (In our scheme, this webpage is a URL on the syndicator, www.S.com, where both ID_{ad} and ID_{pub} are in the URL.) For example, X.html might take the following form:

```
<html><body>
<script language="JavaScript">
//Determine referring webpage r
// (which contains ID_ad and ID_pub):
var r = escape(document.referrer);
//Write HTML to release the coupon γ.gif:
document.write('<img src="http://S.com/'
  + 'γ.gif?ref=' + r + '"/>');
</script> </body> </html>
```

Now when the syndicator's site page with a URL containing ID_{pub} and ID_{ad} references X.html, the syndicator www.S.com receives a request for the resource γ.gif?ref=www.S.com%3fad%3d$\langle ID_{ad}\rangle$%26pub%3d$\langle ID_{pub}\rangle$ (the value of the ref querystring variable in this resource request is the referrer, or page that triggered X.html to load, but encoded so it can appear in the URL). In essence, he receives a request for an image γ.gif, and is provided one querystring-style parameter containing the IDs of the advertisement and publisher. This string conveys the full desired coupon data $C = (\gamma, ID_{pub}, ID_{ad})$.

Remark: In cases where JavaScript is disabled by a client, an alternative approach is possible. An attestor can create not *one* cache cookie, but an array of cache cookies on independently created values $\gamma_1^{(0)}, \ldots, \gamma_k^{(0)}$ and $\gamma_1^{(1)}, \ldots, \gamma_k^{(1)}$. To encode an k-bit publisher value $ID_{pub} = b_1 \parallel \ldots \parallel b_k$, the publisher releases cache cookies corresponding to $\gamma_1^{(b_1)}, \ldots, \gamma_1^{(b_k)}$. Of course, this method is somewhat more cumbersome than use of document-referrer strings, as it requires the syndicator to receive and correlate k distinct cache cookies for a single transaction.

4.4 Freshness

Authentication alone is insufficient to guarantee valid coupon use. It is also imperative to confirm that a coupon is *fresh*, that is, that a client is not replaying it more rapidly than justified by ordinary use.

To ensure coupon freshness, a syndicator may maintain a data structure $T = \{R^{(1)}, \ldots, R^{(r)}\}$ recording coupons received within a recent period of time (as deter-

mined by syndicator policy). A record $R^{(i)}$ can include an authentication value $\gamma^{(i)}$, publisher identity $ID_{pub}^{(i)}$, ad identifier $ID_{ad}^{(i)}$, and a time of coupon receipt $t^{(i)}$.

When a new coupon $C = (\gamma, ID_{pub}, ID_{ad})$ is received at time t, the syndicator can check whether there exists a $C^{(i)} = (\gamma, ID_{pub}, ID_{ad}) \in T$ with timestamp $t^{(i)}$. If $t - t^{(i)} < \tau_{replay}$, for some system parameter τ_{replay} determined by syndicator policy, then the syndicator might reject C as a replay. Similarly, the syndicator can set replay windows for cross-domain and cross-advertisement clicks. For example, if $C^{(i)} = (\gamma, ID_{pub}^{(i)}, ID_{ad}^{(i)})$, where $ID_{ad} \neq ID_{ad}^{(i)}$, i.e., it appears that a given user has clicked on a different ad on the same site as that represented by C, the syndicator might implement a different check $t - t^{(i)} < \tau_{crossclick}$ to determine that a coupon is stale and should be rejected. Since a second click on a given site is more likely representative of true user intent than a "doubleclick," we would expect $\tau_{crossclick} < \tau_{replay}$.

Of course, many different filtering policies are possible, as are many different data structures and maintenance strategies for T.

5 Prototype Implementation

We implemented a prototype of our premium-click scheme. Four websites at separate IP addresses provide a simulated advertiser, publisher, attestor, and syndicator. The web sites are served by Apache 2.0.58, and server-side scripted with PHP 5.1.6. The database for click, ad, and coupon data is MySQL 5.0.26.

Advertiser. The prototype advertiser consists of two fabricated product pages designed as destinations for a user that clicks on a web ad. The only other duties of an Advertiser in the premium clicks system are to submit the ads to the syndicator, and then pay for billed clicks.

Publisher. The prototype publisher is a simple site that embeds ads, served by the syndicator, in iframes. Many widespread advertisement schemes (including Google's AdSense) use this technique; others simply write directly to a publisher's page, submitting their advertisements to the same origin as the publisher, thus making the scheme vulnerable to more click-fraud techniques [4].

Attestor. The prototype attestor is a simple service that provides a login box. When a user of the service provides a valid ID and password, he is provided an internal page that serves a cache cookie to the visitor's browser. This simple HTML file is transmitted from the attestor to the visitor only once after login. Any subsequent requests for the cache-cookie URL are replied to with an HTTP 304 "not modified" response. This forces the browser to use a cached version of the cookie if it exists, and does not provide one to browsers lacking a cached version of the cookie.

The cache cookie served by the attestor references an image hosted on the syndicator. The URL used to request the image is created by JavaScript when the cookie's HTML is rendered, and contains the secret γ (which is generated when the cache cookie is set) as well as the referrer page, i.e., whichever page caused the cache cookie to load. Later, when the cookie is loaded in conjunction with a click, the URL of the referrer will reveal the ID of the publisher and the ID of the advertisement that was clicked.

The attestor needs to create and serve these cache cookies when the user logs in, so additional processing is required. However, creating a secret value takes very little time, and the cookie can be served in a hidden iframe. The result is no difference in experience for the user, and only a trivial amount of work for the attestor's servers.

Syndicator. Of all the entities, the syndicator does the most work. It receives coupons released by the cache cookies (in the form of requested images), verifies the secrets in the coupons, and records clicks. Additionally, each ad click must be directed "through" the syndicator, so it must also serve a transfer page to direct the client's web browser to the advertiser's site. This is an ordinary flow of traffic in ad-serving systems that briefly delegates control to the syndicator who records clicks.

- *Database.* The syndicator hosts a MySQL database to house the advertisement data (content, advertiser's ID, URL), click-through log (each click as it occurs, including advertisement ID and publisher ID), as well as a log of received coupons. Since the released coupon history needs to be saved and searched, we chose to use a database to ease development.
- *Processing Clicks.* When an advertisement is clicked, the client's browser navigates to the syndicator's site, bringing along the advertisement ID and the publisher ID. For example: `http://syndicator/click.php?ad=`x`&pub=`y. The syndicator then records the click, and responds with a web page that causes cache cookies from all attestors to load. (We only implemented one attestor, so one iframe is rendered with its content being the attestor's cache cookie. When there are N possible attestors, N iframes are used.) Attestors' cache cookies not available in the browser's cache are simply not loaded. Coupons are released by the client's browser from any attestor's cache cookies.

- *Receiving Coupons.* Coupons are received by the syndicator in the form of requests for an image called "coupon.gif". When this is requested, it is accompanied by a querystring. For example:
  `http://syndicator/coupon.gif?secret=`γ`&ref=`x.
  The `ref` variable in the query string reflects both the publisher ID ID_{pub} and the advertisement ID_{ad} that was clicked. Receiving this request, the syndicator records the time, secret γ and referrer x in the database, and then serves a tiny image back to the client. HTTP headers are provided that force the client's browser always to request this image, and not load it from cache. The purpose of this process is to ensure the coupons are always freshly delivered, and not loaded from browser cache.

- *Analyzing Clicks.* On the syndicator's click-through page where the attestors' cache-cookie iframes are present, a small delay is forced by JavaScript to allow the coupons transit time, since they load asynchronously in iframes. Immediately following that, the server decides if a click should be classified as "premium." This is done by looking through the coupon database for recently released coupons corresponding to the advertisement that was clicked. Time between when the click was recorded and when the coupons arrived is noted, and only coupons within a pre-set window (τ_{replay}, sixty seconds in our prototype) are considered in determining the premium status. If coupons are present and the secrets are valid, the click is recorded as "premium."[4] Otherwise it is recorded as a general-class click.

  In a production system, click analysis would be done after redirecting the client by adding it to a processing queue.

6 Privacy

In deploying our premium-click scheme with multiple attestors, $\mathcal{A}_1, \ldots, \mathcal{A}_q$, it would be natural for a syndicator to share a unique key k_i with each attestor $\mathcal{A}_i$. Given such independent attestor keys $\{k_i\}$, though, a coupon created by $\mathcal{A}_i$ conveys and therefore reveals the fact that a user has visited the Web site of $\mathcal{A}_i$. Observe, however, that in our scheme a publisher triggers the release of a coupon from the browser of a visiting user, but does not see the coupon. The syndicator receives the coupon, but does not directly interact with the user. In effect, the syndicator receives the coupon *blindly*. While the syndicator does learn the IP address of the user, this is information that is typically already available: The only additional information that the syndicator learns is whether or not the user has received an attestation. Thus, coupons naturally decouple information about the browsing patterns of users from the identities and browsing sessions of users. This is an important, privacy-preserving feature.

Such decoupling occurs in the case when ads are outsourced, that is, when the syndicator and publisher are separate. When the syndicator and publisher are identical, i.e., when a search engine displays its own advertisements, coupons may be linked to users, and therefore leak potentially sensitive information. A couple of privacy-enhancing measures are possible. To limit the amount of leaked browsing implementation, our scheme may employ a multiple-coupon technique discussed in depth in Appendix A. Alternatively, attestors may share a single key k (or attestors may have overlapping sets of keys). In this case, a MAC does not reveal the identity of the attestor that created it. If a coupon $\gamma = m \parallel MAC_k(m)$ is created, as we propose, with a random nonce m, then it conveys no information about a user's identity. In principle, however, it would be possible for an attestor to embed a user's identity in m, thereby transmitting it to the syndicator. This transmission could even be covert: A ciphertext on a user's identity, i.e., an encryption thereof, will have the appearance of a random string. Proper auditing of the policy and operations of the attestor or syndicator would presumably be sufficient in most cases to ensure against collusive privacy infringements of this kind.

As an alternative, m might be based on distinctive, but verifiably non-identifying values. For example, m might include the IP address[5] and/or timestamp of the client to which an attestor issues a coupon—perhaps supplemented by a small counter value.[6] A client could then verify that m was properly formatted, and did not encode the user's identity. Of course, $MAC_k(m)$ itself might then embed the user's identity. It is possible, however, to eliminate the possibility of a covert channel in the MAC by periodically refreshing k and publicly revealing old values.

Remarks:

- There are good business motivations for attestors not merely to validate, but also to classify users. For example, an retailer might not merely indicate in a coupon that a client has spent enough to justify validation, but also provide a rough indication of much the client has spent ("low spender," "medium spender," "profligate"). Advertisers might then be charged on a differential basis according to the associated, perceived value of a client. Such classification would create a new dimension of privacy infringement. In the outsourcing case, where coupons

are decoupled from user identities, this approach might meet with user and regulator acceptance. In the case where the syndicator publishes advertisements, and coupons are linked to users, privacy is of greater concern. As advertisers will necessarily learn the syndicator's differential pricing scheme, there will be at least some transparency.

- In principle, public-key digital signatures offer more flexible privacy protection for coupon origins than MACs. For example, group signatures, e.g., [1], permit the identity of a signing attestor to be hidden in the general case, but revoked by a trusted entity in the case of system compromise. In a high traffic advertising system, however, public-key cryptography would be prohibitively resource-intensive.

7 Security Analysis

Without possession of an attestor key, an adversary cannot feasibly forge new coupons, thanks to our use of MACs. An adversary could still bypass our scheme in several ways:

- **Direct publisher fraud:** Using a slight modification of the proposed solution, the publisher could cause release of coupons even when users do not click on ads.
- **Indirect publisher fraud:** A dishonest Web site could re-direct users to the publisher's site.
- **Malware-driven clicks:** A virus or widely spread Trojan could either surreptitiously direct a user's browser to a Web site and simulate a click or else steal a coupon from the browser for use on another platform.

All of these attacks are possible in existing click-fraud schemes. The various techniques used to address them today are equally applicable to premium clicks. For example, a syndicator can direct its own client machines to a publisher's site to determine if the publisher is generating fraudulent clicks. Indeed, our premium-click scheme makes detection of misbehavior easier, as it permits a syndicator to "mark" a client coupon and therefore directly monitor the traffic generated by the client and even detect the emergence of stolen coupons.

An adversary can also try to exploit the special characteristics of our scheme as follows:

- **Posing as an attestor:** An adversary might either establish itself as an attestor or compromise the key of an existing attestor. If the syndicator sets appropriate policies for creating attestors, then it should be difficult for an adversary to pose as one. Attestors are likely, in any case, to be a more exclusive class of Web site than publishers or even advertisers. Moreover, in the case where MAC keys are attestor-specific, the syndicator can individually monitor the traffic generated by each attestor, making fraud detection easier.
- **Compromise of an attestor key:** An adversary can attempt to learn the MAC key of an existing attestor. The difficulty of this form of attack depends on the security of the attestor's Web site. MAC keys for premium clicks may be protected using many of the same measures employed to secure SSL keys and other cryptographic secrets.
- **Coupon harvesting:** An adversary could harvest coupons from attestors by creating accounts or clients that meet their validation criteria. By establishing appropriate policies for validation by its attestors, the syndicator can attempt to attach a financial cost to this form of fraud in excess of the gains that a fraudster might reap from it.

Auditing

Since the syndicator is ultimately in control over deciding which clicks should be considered "premium" (and earns more when clicks *are* premium), publishers and advertisers may accuse the syndicator of improperly inflating the percentage of clicks considered premium. To solve this problem, an additional entity called an *auditor* can be contracted to watch the coupons that are released, and verify the premium-status judgement of the syndicator. The auditor would not be rewarded based on click traffic, so it would have no incentive to inflate or deflate the number of premium clicks from those that are legitimate.

The cache cookies set by attestors can be crafted so that, when an advertisement's URL is clicked, the coupon $C = (\gamma, ID_{pub}, ID_{ad})$ is released both to the syndicator *and* to the auditor who maintains an independent database. When the syndicator's numbers are contested, the coupons recorded by the auditor can be used to recompute the number of premium clicks for a given advertisement or publisher, and compared to the syndicator's calculation.

8 Conclusion

In contrast to today's heuristic filtering methods for eliminating "bad" clicks, our premium-click scheme relies on a foundation of cryptographic authentication to validate "good" clicks. Premium clicks are by no means a cure-all for fraud, and are themselves subject to attack. The

value of premium clicks lies in the way that they provide new, cryptographically authenticated visibility into click traffic, and thus a new, stronger platform for combating click fraud.

While premium clicks could in principle supplant current filtering schemes entirely, they are attractive in that they can be deployed in a complementary fashion alongside existing systems. We have proposed a new advertising model in which advertisers pay a higher charge for premium clicks. We believe that such a scheme might be launched experimentally by a syndicator with minimal impact on existing business and then expanded as its success warrants. Thus premium clicks promise offer not only a new approach to click fraud, but one with a practical path to fruition.

References

[1] ATENIESE, G., CAMENISCH, J., JOYE, M., AND TSUDIK, G. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology-Crypto '00* (2000), M. Bellare, Ed., Springer. LNCS vol. 1880.

[2] BELLARE, M., CANETTI, R., AND KRAWCZYK, H. Keying hash functions for message authentication. In *CRYPTO '96: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology* (London, UK, 1996), Springer-Verlag, pp. 1–15.

[3] EBAY. www.ebay.com, Accessed 31 January 2007.

[4] GANDHI, M., JAKOBSSON, M., AND RATKIEWICZ, J. Badvertisements: Stealthy click-fraud with unwitting accessories. In *Anti-Phishing and Online Fraud, Part I Journal of Digital Forensic Practice, Volume 1, Special Issue 2* (November 2006).

[5] JUELS, A., JAKOBSSON, M., AND JAGATIC, T. Cache cookies for browser authentication (extended abstract). In *IEEE Symposium on Privacy and Security* (2006), pp. 301–305.

[6] RSNAKE. Stealing mouse clicks for banner fraud. http://ha.ckers.org/blog/20070116/stealing-mouse-clicks-for-banner-fraud/, January 2007.

[7] STAMM, S., RAMZAN, Z., AND JAKOBSSON, M. drive-by pharming, 2006. Technical Report, http://www.cs.indiana.edu/pub/techreports/TR641.pdf.

[8] TSOW, A. Phishing with consumer electronics – malicious home routers. In *In Models of Trust for the Web, a workshop at the 15th International World Wide Web Conference (WWW2006)* (2006).

[9] TSOW, A., JAKOBSSON, M., YANG, L., AND WETZEL, S. Warkitting: the drive-by subversion of wireless home routers. In *Anti-Phishing and Online Fraud, Part II Journal of Digital Forensic Practice, Volume 1, Special Issue 3* (November 2006).

[10] TUZHILIN, A. The Lanes gifts v. Google report, 2006. Independent evaluators assessment of quality of Googles click-fraud filtering methods. Accessed 31 January 2007 at http://googleblog.blogspot.com/pdf/Tuzhilin_Report.pdf.

Notes

[1] This research was performed by the author at RavenWhite Inc.

[2] The reason for having the syndicator trigger coupon-release is twofold: (1) To prevent JavaScript-based click automation, ads today are often rendered inside an iframe whose source is loaded from the syndicator's site and (2) To eliminate the need for JavaScript on the publisher's site.

[3] An authenticator could create a list X of random codes and transfer it to the syndicator via a backchannel, but this would not be efficient (and would also eliminate some of the privacy properties we would like to achieve).

[4] In a multi-attestor environment, where clients may carry and release multiple coupons, the syndicator needs some mechanism to determine which coupons correspond to a given client. A simple option is to attach a fresh, random number (nonce) to the links in each rendered advertisement. The nonce will attach itself to all of the coupons that a client releases in a given click.

[5] Inclusion of an IP address in a coupon also has some security benefits. In the case where the syndicator publishes its own ads, it can check that a client's presented IP address is consistent with the IP address in the coupon, e.g., it originates with the same service provider.

[6] A counter might still embed a covert channel, but, if the size of the channel might be made small enough to alleviate the problem of privacy infringement significantly.

A Multiple Attestors

User privacy in our premium-click scheme depends upon how the value γ is formed, and on the number and content of the coupons cached in a user's browser. Let us now therefore consider a system with multiple attestors, $\mathcal{A}_1, \ldots, \mathcal{A}_q$. Each attestor $auth_i$ shares a key k_i with the syndicator. We now describe the technical challenges that arise with multiple attestors.

Multiple coupons. The first problem we encounter in a system with multiple attestors is the difficulty of managing multiple cache cookies across different domains. A cache-cookie system can involve caching of a set of j different webpages $X_1, X_2, \ldots, X_j$ in a given user's browser, each webpage serving as a *slot* for a distinct cache cookie. Two difficulties arise, however. The first is that a site seeking to release a set of cache cookies (i.e., the publisher) cannot determine what slots in a user's browser actually contain cache cookies. The only way for the publisher to release all cache cookies is to call all j webpages. The second is that a site seeking to set a cache cookie, i.e., an attestor, cannot determine if a given slot has been filled. If the attestor plants a cache cookie in a slot that already contains one, the previously planted cache cookie will be effaced.

The simplest way to circumvent these difficulties in our premium-clicks scheme is to manage only a single slot, that is, to maintain only a single cache cookie in a given user's browser. Only the cache cookie planted most recently by an attestor will then persist. Provided that the syndicator regards all attestors as having equal authority

in validating users, this approach does not result in any service degradation.

If, however, the syndicator desires the ability to harvest multiple coupons, then attestors must use multiple slots. One possible approach is to maintain an individual slot for each attestor, i.e., to let $j = q$. If the number of attestors is small, this may be workable. Alternatively, attestors may plant coupons in random slots, sometimes supplanting previous coupons, or subsets of attestors may share slots. The syndicator might, for example, assign different weight to attestors, according to the anticipated reliability of their attestations; attestors with the same rating might share a slot.

Keying. One approach to management of attestor keys is to assign an identical key k to all attestors, i.e., let $k_1 = k_2 \ldots = k$. While this approach has the merit of simplicity, it has the disadvantage of rendering tracing and key-revocation difficult.

It is preferable, therefore to create attestor keys $\{k_i\}$ in an independent manner. In this case, a coupon $\gamma = m \parallel MAC_{k_i}(m)$ is cryptographically bound to the attestor that created it. That is, only attestor $\mathcal{A}_i$, with its knowledge of k_i, can feasibly create γ of this form. To enable the syndicator to determine the correct key for verification of the MAC, the coupon must be supplemented with i, the identity of the authenticator. For example, we might let $m = i \parallel r$, where r is a random nonce.