*This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.*

# Addressing Click Fraud in Content Delivery Systems

Saugat Majumdar, Dhananjay Kulkarni and Chinya V. Ravishankar
University of California, Riverside, CA 92521
{smajumdar, kulkarni, ravi}@cs.ucr.edu

***Abstract*—Mechanisms for data access and payment are central to the success of content delivery systems. However, not much attention has been paid to the issues of dishonest intermediaries (brokers) or client collusion with dishonest brokers. We propose protocols to verify broker honesty for data accesses under standard security assumptions in such systems. Analytical and experimental results show that our protocols are robust against replay and fabrication attacks, and are consistently able to identify broker dishonesty.**

## I. Introduction

On-line information and data service is a growing industry. Stock exchanges, news services, and on-line vendors such as Yahoo, already market stock quotes, news, and music, respectively, on the Internet. Roles are also becoming specialized. Publishers may have data domain expertise, but may not be able to disseminate data or manage clients efficiently.

Therefore, an ancillary industry of data *brokers* has developed in parallel with the content creation industry. Brokers may maintain servers to enhance data delivery quality, manage subscriptions, provide anonymity guarantees, and support different payment options for clients and publishers. Examples of brokers or intermediaries can include Akamai and C&W, which provide enhanced data dissemination features.

Current systems typically require publisher to trust brokers to behave honestly, though such trust may not always be warranted. We do not assume that brokers are honest, and propose methods to detect broker dishonesty.

*Click inflation*, a topic of current interest, can be caused by broker dishonesty or neglect, with reports suggesting that up to 20% of reported clicks may be fraudulent. Major players such as Yahoo and Google have already been settling significant allegations [1] of click fraud.

As the content brokerage industry grows, so will the need for security protocols to guard against broker dishonesty. Work exists on pricing techniques in this domain [15], [24], but such work tends to assume honest brokers and clients. This assumption is increasingly becoming untenable. We propose schemes to alert publishers to broker dishonesty.

### A. Content Delivery Systems

A Content Delivery System (CDS) is a networked system of computers cooperating transparently to deliver content to end-users. We consider a CDS (see Figure 1) in which *publishers* produce information and disseminate information to *clients* through a supporting network of *brokers*. Clients register and interact with brokers, and never directly with publishers. The broker network maintains a set of servers optimized for fast data search and delivery, the details of which are not relevant to publishers or clients.

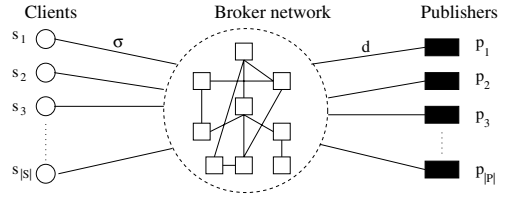



Fig. 1. Schematic Model of A Content Delivery System

*1) Payment Models:* We classify content delivery systems based on the payment mechanism. In the **broker-payee model**, the publisher *pays the broker* based on the number of times the published data is accessed. For example, consider a service that allows advertisers (" publishers") to post advertisements on websites owned by a web-host ("broker"). Internet users ("clients"), such as online shoppers click on the advertisements and are directed to target sites for more information on the product. The web-host monitors the web-clicks and charges the advertiser based on the number of web-clicks seen by the advertisement.

In the **publisher-payee model**, the broker *pays the publisher* an amount proportional to the number of accesses to the published data. This amount is an agreed-upon percentage of the total payments made by the clients for accessing the published data. For example, a news agency may distribute articles through a broker, who generates readership and distributes articles. Readers register with the broker, read the articles online, and pay the broker per article read. The broker shares a certain fraction of these revenues with the publisher.

### B. Broker-Driven Click Fraud

Since payments in both models depends on the number of client accesses to published data, the broker has an incentive to report a wrong number of accesses. In our model, any data access that results in data delivery corresponds to a *click*. Using this analogy, we will have *click fraud* at the broker level if it reports a wrong count of the number of accesses (or clicks) to the published data.

In the broker-payee model, a broker can cheat the publisher by reporting an *overcount* for the number of data accesses. In the publisher-payee model, the broker can cheat by reporting

an *undercount*. In practice, the payment function is likely to be linear or piecewise linear. Hence, to increase profit by a factor of $k$, the broker would have to report a count different by a factor of at least $k$.

We have found no previous work that is resilient to broker and client dishonesty. Broker dishonesty is briefly mentioned in [24], but not solved. We believe our work is first in defining the problem setting and to provide a solution for both payment models.

### C. Our Approach

We address the issue of reporting incorrect counts by requiring the broker to *report* every data access request to the publisher, who will *validate* the report. We must also guarantee that client identities are not disclosed to publishers without permission, and ensure that our protocols are efficient.

*1) Stable Bloom Filter-based Solution for Broker-payee Model:* We track report replays by using an extension of the classical Bloom Filter [7], called the Stable Bloom Filter [13], at the publisher. Hits in the Stable Bloom Filter signal replays. We eliminate false positives through a challenge-response protocol between the publisher and the client. We provide clients an incentive to give up anonymity, and propose a probabilistic approach for identifying fabricated reports. Our solution identifies most replays and reduces false positives, at a very low storage cost.

*2) Challenge-Response-based Solution for Publisher-payee Model:* We run a challenge-response protocol between the client and the publisher, such that all the legitimate data deliveries are successfully reported to the publisher. This solution rewards clients that initiate access notification, and hence is able to identify all the legitimate data accesses.

*3) Solution to Address Client-Broker Collusion:* Publishers in the broker-payee model guard against collusion by tracking the number of reports received for any client, tagged anonymously through a public key received with reports.

When the number of colluding clients is not large, a broker may generate large fraudulent profits only if each colluding client participates in hiding a large number data accesses from the publisher, and becomes subject to detection. Clients in publisher-payee model have low incentive to participate in such collusions.

## II. Related Work

Broker dishonesty is mentioned in [24], [15], but no solutions are discussed. Work in [23] proposes a pricing technique for publish-subscribe systems, but does not address cheating by a broker. Traditional payment schemes [3], [6], [11], [16] do not go through a broker network, hence they are inapplicable in our problem setting. The problem of *click-inflation* introduced in [4] is a related problem, but does not address the security requirements we consider.

Bloom filters have been used in [18] to detect duplicate clicks. This work is not useful because brokers could cheat by replaying the entries that were deleted from the Bloom Filter.

| | | | | |
|---|---|---|---|---|
| $B$ | Broker network | | KMS | Key Management Service |
| $p$ | Publisher | | $sig_s()$ | Signature of $s$ |
| $b_m$ | Master broker | | $\pi_p(\pi_s)$ | $p(s)$'s Public key |
| $s$ | Client | | $cert(,)$ | Certificate |
| $\widehat{s}$ | Dishonest $s$ | | $(d, I_d)$ | (Data,Identifier) |
| $\widehat{b}_m$ | Dishonest $b$ | | $c, \rho$ | challenge,response |
| $\sigma$ | Predicate | | $sig_p()$ | Signature of $p$ |
| $h_i()$ | Hash Function | | $SBF_p$ | SBF at $p$ |
| $m$ | Size of $SBF_p$ | | $r_d(r_s)$ | Record for $d(s)$ |
| $l$ | # hash funcs | | $anon$ | Anonymity Flag |
| $count$ | Count at $\widehat{b}_m$ | | $count_p$ | Count at $p$ |

Fig. 2. Our Notation

## III. Requirements in Content Delivery Systems

Content delivery systems (CDS) must typically satisfy at least the *information integrity* and *client anonymity* requirements [24].

**Definition 1.** *A CDS maintains* **information integrity** *if it delivers each data item to clients with the same information content as it had when it was published.*

**Definition 2.** *A CDS maintains* **client anonymity** *if it leaks no information about client identities to publishers.*

Information integrity is a data correctness guarantee. Client anonymity preserves privacy and prevents malicious publishers from sending inaccurate data to selected clients.

## IV. Protocols When Everyone is Honest

Consider a system with a set of publishers $P$, a set of clients $S$, and a broker network $B$. We assume that public keys are managed by a key management service (KMS).

For simplicity of exposition, we will assume that only one "master" broker $b_m \in B$ holds each data item $d$ to be returned in response to a client request. We will describe the CDS in terms of the four operations `register`, `publish`, `access` and `count` in Figure 3.

*1) The Register Protocol:* This protocol allows a client $s$ to register with the broker network $B$. Upon successful authentication and registration, a record $r_s$ is created within $B$ holding the client's public key $\pi_s$, a certificate $cert(s, \pi_s)$ binding the client identity to $\pi_s$, and an anonymity flag $anon$. This flag is set to $false$ if the client is willing to disclose its identity, and to $true$ otherwise. Master brokers have access to these records.

*2) The Publish Protocol:* This protocol allows a publisher $p$ to publish a data item $d$ through the broker network $B$, and creates a new record $r_d$ at the master broker $b_m$ that manages data item $d$. The record $r_d$ has the fields $\{p, I_d, d, count, sig_p(d), cert(p, \pi_p)\}$, where $p$ is the publisher identifier, $I_d$ is a data item identifier, $d$ is the data item, $count$ is the number of times $d$ has been delivered to clients, $sig_p(d)$ is the publisher's signature on $d$, and $cert(p, \pi_p)$ is a KMS certificate on the publisher's public key $\pi_p$.

*3) The Access Protocol:* This protocol allows a registered client $s$ to request and retrieve from $B$ a data item $d$ satisfying a predicate $\sigma$. This request is propagated within the broker network $B$, until broker $b_m$ delivers $d$ to $s$, and increments the value of $count$ in the record $r_d$.

The `access` protocol guarantees information integrity and client anonymity. A client $s$ can check the integrity of the data by validating the signature $sig_p(d)$ using the public key in $cert(p, \pi_p)$. Client anonymity is maintained if the broker $b_m$ does not disclose any information that identifies $s$ to the publisher.

*4) The Count Protocol:* The `count` protocol allows a broker and publisher to reconcile their $count$ values. The publisher $p$ requests $b_m$ for $count$ corresponding to a particular data item $d$. The broker $b_m$ responds to $p$ with the value of $count$ stored in $r_d$.

The integrity of the messages exchanged between the clients and brokers, and between publishers and brokers is maintained by using digital signatures [22].

`Register(s, B,` $\pi_s$`,` $cert_s$`, anon) :`
G1) $B$ creates a record $r_s = \{s, \pi_s, cert_s, anon\}$.

`Publish(p,` $b_m$`, d) :`
P1) $p$ creates unique identifier $I_d$.
P2) $p$ sends $\{p, I_d, d, sig_p(d), cert(p, \pi_p)\}$ to $b_m$.
P3) $b_m$ creates a record $r_d = \{p, I_d, d, sig_p(d), cert(p, \pi_p), count\}$ with $count = 0$.

`Access(s, B,` $\sigma$`) :`
A1) $s$ sends predicate $\{\sigma\}$ to $b_m \in B$
A2) $b_m$ returns $\{d, sig_p(d), cert(p, \pi_p)\}$, for $\sigma(d) = T$.
A4) $b_m$ updates the value of $r_d.count$.

$Count(p, b_m, I_d)$ :
C1) $p$ sends a request $\{I_d\}$ to $b_m$.
C2) $b_m$ sends $r_d.count$ to $p$.

Fig. 3. Protocols when Everyone is Honest

## V. Problem Formulation and Assumptions

**Definition 3.** *A broker is* **dishonest** *if it either reports a wrong count for the number of accesses to the published data, or colludes with clients to fool publishers into accepting the wrong count for published data.*

**Definition 4.** *A content delivery system maintains* **count integrity** *for a data item $d$ if $d$'s publisher can correctly determine the number of times $d$ was accessed by the clients.*

This requirement is vital because the payment between the publisher and the broker proportional to the number of accesses to the data. We will address the following problem:

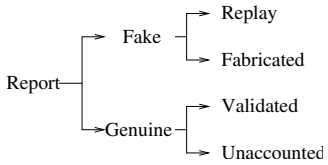



Fig. 4. Types of Report

**Problem Statement.** *Given the set of clients $S$, set of publishers $P$ and the broker network $B$, maintain count integrity while ensuring information integrity and client anonymity.*

**Assumption 1.** *Brokers may be dishonest.*

**Assumption 2.** *Brokers and publishers do not collude.*

**Assumption 3.** *Clients do not deny an access after receiving the requested data item from the broker.*

**Assumption 4.** *A broker discloses a client's identity only after seeking permission from the client.*

**Assumption 5.** *All communication links are reliable.*

## VI. Using Reports to Maintain Count Integrity

We maintain count integrity by having the broker and the publisher each maintain a count of the number of accesses for each data item. We require the broker to report every access request to the publisher to enable it to maintain this count.

**Definition 5.** *A* **report** *is a claim by a broker that a client has performed an access $r$ to a data item $d$. A report is* **genuine** *if request $r$ caused the broker to deliver data $d$ to the client. A report is* **fake** *otherwise.*

### A. Verifiable Reporting in Broker-payee

We classify the reports as shown in Figure 4. In the broker-payee model, a broker may *replay* a previous report, or *fabricate* a new report to fool the publisher. We require each report to be *verifiable*. Verifiable reports are validated by the publisher as being fake or genuine.

A verifiable report includes a client signature on the value $(I_d|p)$, where $p$ is the publisher and $I_d$ is the identifier of the data item $d$. A client will include this signature with the request it sends to a broker, and the broker forwards it to the publisher, and claims payment. If the publisher is able to verify the signature on a report, it becomes a validated report.

We use a probabilistic signature scheme [17], [14], [19], so that no two signatures from a client for a given data item are the same. Broker replay of signatures is deterred as long as the publisher can detect duplicate signatures.

### B. Reporting in Publisher-payee

In the publisher-payee model, the dishonest broker may generate undercounts by not reporting the data accesses to the publisher and hence, resulting in *unaccounted* reports.

### C. Our Policy to Maintain Count Integrity

Publishers deter dishonest brokers with the following policies. We assume that penalty can be imposed on a dishonest broker by executing the function $Penalty()$.

**Validation-based Payment (Broker-payee Model):** *Publishers make payments only for validated reports.*

We argue that this policy maintains count integrity as long as fabricated or replayed reports fail validation at the publisher. Each client sends a correct signature with each access, so the broker will forward it and claim payment, having no incentive to fabricate a report in this case. These reports are correctly validated and counted at the publisher. For good signature schemes, we can discount the possibility of the broker fabricating client signatures, so that fabricated reports will be detected and discarded by publishers. If publishers can also detect replays, we will have ensured count integrity.

**Reward-based Notification (Publisher-payee Model):** *Publishers reward clients for participating in the report notification process.*

We argue that this policy maintains count integrity as long as the client can initiate the notification and monitor the broker behavior during the notification process. The clients are willing to participate in the notification process to gain rewards.

## VII. Stable Bloom Filter-based Solution for Broker-payee Model

We propose a technique using Stable Bloom Filter (SBF) [13] to stop replay and fabrication of reports. Each publisher keeps track of the signatures received from the broker by entering them into a SBF. The SBF mechanism consists of a set of hash functions $h_1(), h_2(), \cdots, h_l()$ with output range $\{0, 1, \cdots, m\}$, and a vector $v$ of $m$ cells, all initialized to 0.

Figure 5 shows the operations defined on a Stable Bloom filter. The `insert` operation applies the hash functions to the input, and sets each of the corresponding cells to $M$, where $M$ is the maximum value that can be assigned to a cell. Our Validation-based Payment Policy is implemented by operation `is_member` in Figure 5, which verifies if a received signature $sig_s(I_d|p)$ corresponds to a replayed report. Finally, the operation `rand_decr` selects $t$ cells at random, and decrements them. The execution of `rand_decr` after every `insert` and `is_member` limits the false positive rate, as shown in [13].

It is shown in [13] that when distribution of inputs does not change over time, the policy of randomly decrementing entries in the SBF causes the FP rate to converge very quickly to a constant. This state of the SBF is called the *stable state*.

### A. Proactive Approach to Stop Replay of Reports

A cell is set to $M$ whenever a value hashes into it, but there is a small probability that this cell will be selected for decrement $M$ times before any other input hashes to it. This cell may possibly get cleared, causing a false negative (FN).

*insert($\gamma$):* Set $v[h_1(\gamma)], v[h_2(\gamma)], \cdots, v[h_l(\gamma)]$ to $M$.
*is_member ($\gamma$):* If any of $v[h_1(\gamma)], v[h_2(\gamma)], \cdots, v[h_l(\gamma)]$ is 0, return $false$, else $true$.
*rand_decr():*
1) Randomly select $t$ distinct indexes $\{i_1, \ldots, i_t\}$.
2) If $v[i_j] \geq 1$ then $v[i_j] = v[i_j] - 1,\ j = 1, 2, \ldots, t$.

Fig. 5. Stable Bloom filter $v$ with hash functions $h_1(), \cdots, h_l()$

The broker knows that the `rand_decr` operation randomly selects and decrements $t$ cells. In principle, a broker can replay a report if it can guess which cells have been cleared. To prevent this possibility, the $t$ cells chosen in each execution of `rand_decr` are kept secret. Consequently, successive states of the SBF are also kept secret.

We prevent replay of reports by forcing the broker to guess which cells are cleared. In Section VIII-C, we bound the probability with which a dishonest broker can guess FNs.

### B. Reactive Approach to Identify False Positives

We call it a *hit* if the `is_member` operation returns *true*. To determine whether the hit is due a false positive, the publisher requests the client identity, and presents a challenge to which the response can only be determined by a valid client. If the client responds correctly, the publisher considers the report to be genuine, and labels the hit as a false positive. Hence, the publisher is able to successfully verify replays and false positives that are signaled by a hit.

Since this protocol requires knowledge of client identity, it is most appropriate when the number of false positives in the SBF are expected to be low. The parameters, $m$, $l$ and $t$ can be set to minimize false positives using prior knowledge of the access request volume.

### C. Probabilistic Identification of Fabricated Reports

A broker is required to include a client signature in a report, but a publisher has no way of knowing whether this signature is genuine. A dishonest broker may generate false public-secret key pairs and create false signatures. If any signature is challenged, it can respond with a public key matching the signature in question.

Consequently, signatures could never be verified using only information from the broker. Any public key supplied by the broker must be tied to an identity that can be verified by a trusted third party. Thus, clients must give up anonymity.

*1) Incentive for Revealing Identity:* Although clients prefer anonymity, it is fortunately quite routine in commerce for some clients to give up anonymity in exchange for monetary incentives. Retail discount cards are the best example, where customers permit tracking of their identities and purchases in exchange for a discount on the purchased items.

It is easiest to model the incentive for a client to give up anonymity as coming from the broker, who may, in turn, be compensated by the publisher. The broker is not to reveal client

```
Access(s, B, σ) :
  A1-A4) Execute steps A1–A4 in Figure 3
  A5) s sends {sig_s(I_d|p), cert(s, π_s)} to b_m
  A6) b_m services request, sends sig_s(I_d|p), to p
  A7) If (is_member(sig_s(I_d|p))) then p initiates
    the Identify_Replay(s, b, p) protocol
  A8) p initiates Identify_Fabri(s, b, p) protocol
```

Fig. 6. SBF-based Protocol for Ensuring Count Integrity

```
Identify_Replay(s, b, p) :
  R1) p executes
        rand_decr();
        insert(sig_s(I_d|p));
  R2) p requests the b_m for the public key of s.
  R3) If anon = 1, b_m sends cert(s, π_s) to p
      else sends null to p.
  R4) If p receives null, then p executes
        ++count_p; HALT
  R5) p executes
        M_r = Rand_string();
        c = E_s(M_r);
  R6) p sends {c, sig_p(m̃)} to b_m, where
      m̃ = c|sig_s(I_d|p).
  R7) b_m forwards {c, sig_p(m̃)} to s.
  R8) If sig_p(m̃) is an invalid signature then s sends
      {null} to p and executes
        Penalty(b_m); HALT
      else s sends ρ = D_s(c) to p and HALT.
  R9) If ρ = M_r then p executes
        ++count_p; HALT
      else p executes
        Penalty(b_m); HALT
```

Fig. 7. Identify_Replay Protocol

```
Identify_Fabri(s, b, p) :
  F1) p executes
      flag = Set_Flag(s, q); /*Returns 1 with
        probability q and 0 with probability (1 − q).*/
  F2) If flag = 0, then p executes
      ++count_p;
      rand_decr();
      insert(sig_s(I_d|p)); HALT
  F3) If flag = 1, then
   3.1) p sends {sig_s(I_d|p), flag} to b_m.
   3.2) If anon = 0 then b_m sends cert(s, π_s) to p
     else sends null to p.
   3.3) If p receives null then goto Step 3.5.a.
   3.4) p verifies the validity of sig_s(I_d|p) by using
     public key π_s of s.
   3.5) If sig_s(I_d|p) is valid then p executes
     a)++count_p;
     rand_decr();
     insert(sig_s(I_d|p)); HALT
   3.6) If sig_s(I_d|p) is invalid then p executes
     Penalty(b_m); HALT
```

Fig. 8. Identify_Fabri Protocol

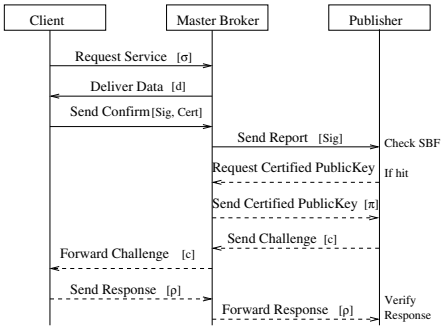



Fig. 9. Sequence Diagram for SBF-based Protocol

identities without client's permission. We do not address the "bad world" scenario when brokers and publishers collude to compromise client identities.

*2) A Probabilistic Approach:* Since it is cost prohibitive to verify every report, the publisher chooses some small fraction of reports to verify. The publisher verifies each arriving report at random with a probability $q$, that can be tuned by the publisher. The publisher requests the broker for the certified public key $\pi_s$ and the KMS certificate $cert(\pi_s, s)$ of the client $s$, and verifies the signature $sig_s(I_d|p)$. Thus, fabricated reports are detected when broker discloses the client identity.

Of course, dishonest brokers may fabricate reports, and when challenged, simply claim that the client has chosen to remain anonymous. We addresses this difficulty by having the publisher monitor the frequency of such claims of anonymity. If this frequency is suspiciously high, the publisher can take additional measures.

### *D. The SBF-based Protocol*

Our protocol is presented in Figure 6. The sequence of message exchanged in the protocol is shown in Figure 9.

When a publisher receives signature $\{id, sig_s(I_d|p)\}$ with a report, it invokes the `is_member` on the signature. If this operation returns $true$, the publisher can choose to flag this report to be a replay, if the SBF parameters were chosen to make the steady-state false positive rate sufficiently small. Alternatively, it can execute a challenge-response protocol with the client to check if the hit is a false positive. If `is_member` returns $false$, the publisher checks whether the report may be a fabrication, as in Section VII-C.

Each execution of `insert` is accompanied by an execution of `rand_decr`. Replayed reports and the genuine reports are both inserted into the SBF. Inserting a replayed report in the

SBF, prevents this report from being a FN for at least $M$ consecutive `rand_decr` operations.

## VIII. Analysis of The SBF-based Protocol

We argue that a dishonest broker can do no better in guessing a FN than an honest broker. We will place the weakest constraints on the broker, allowing him full freedom to manipulate the SBF at the publisher, thereby showing optimal resistance to the threat of FNs.

### A. Limiting Brokers from Spoofing Public Keys

If we allow users to generate their own public keys, a broker may fake (or *spoof*) public keys, and sign fabricated reports with spoofed keys. We will hence require all entities to obtain key pairs from a trusted authority, as in [8]. To prevent the broker from obtaining too many key-pairs, the trusted authority can keep a count of the number of key-pairs distributed to each entity.

### B. Optimal Resistance to Guessing False Negatives

Let Bob claim to be able guess an FN correctly. Let Alice be assigned to evaluate Bob's performance. Alice holds an SBF, with all the cells initially set to 0. To provide maximum advantage to Bob, we assume that he is the only source of `is_member` and `insert` operations arriving at Alice. Alice acts as a publisher who follows the rules in Protocol 6.

*Rule 1:* The $t$ cells selected in the `rand_decr` operation are chosen from a uniform random distribution, and the selection is kept secret.

*Rule 2:* Alice performs `rand_decr` before executing each `insert` request from Bob.

*Rule 3:* For every `is_member(a)` request, Alice executes `is_member(a)`, returns the result to Bob, and executes `rand_decr` and `insert(a)`, in that order.

At time $t_1$, let Bob make a `insert(r)` request to Alice, where $r$ is chosen by Bob. Let $c_b = \{c_1, c_2, \ldots, c_l\}$ be the set of cells set to $M$ by `insert(r)`. Let Bob specify $r$ to be the report for which it will correctly guess a false negative.

Bob continues making his operation requests to Alice. After $U$ operations, let Bob claim that $r$ is a FN with a non-negligible advantage over any honest broker.

*1) Argument for Optimal Resistance:* For Bob to guess that $r$ is an FN after $U$ operations, he must be able to guess that at least one of the cells in $c_b$ is 0. Since all cells in $c_b$ were set, one them could have reached 0 only due to repeated executions of `rand_decr`. However, the $t$ cells decremented in each execution of `rand_decr` were chosen independently of Bob's requests and the states of the SBF. Bob can have no advantage over an honest broker in guessing if any cell in $c_b$ is 0. Hence, a dishonest broker does not have any advantage over a honest broker in guessing a FN.

### C. Probability of Guessing False Negatives

Let us assume that a dishonest broker $\widehat{b}_m$ knows all the reports ever entered in the SBF, the hash functions $h_i()$ used, the size $m$ of the vector $v$, and the value of $M$. Given a cell in the SBF, $\widehat{b}_m$ can easily keep track of which reports causes that cell to be set to $M$.

Let cell $c$ have been set to $M$ during some `insert` operation. Let $\phi(c)$ be the number of operations for which $c$ has not been set. Since $\widehat{b}_m$ knows all reports, it can determine $\phi(c)$ by counting the number of operations for which $c$ has not again been set.

The dishonest broker $\widehat{b}_m$ can target the cell $c$ for which $\phi(c)$ is highest, since this cell is the most likely to have been decremented to 0 by the `rand_decr` operations. (We note that $v[c]$ can be 0 only if $\phi(c) \geq M$.) It can then replay a report that causes this cell to be set, resulting in a false negative.

Since hash function $h_i()$ sets one of the $m$ cells in the SBF, the the probability that $c$ is set by any one of the $l$ hash functions is $l/m$. Similarly, since $t$ cells are chosen randomly by `rand_decr`, $c$ is decremented with probability $t/m$.

Let $\phi(c) = u$ and $\alpha = t/m$. The probability that $v[c]$ is zero is $\Pr[v[c] = 0 \,|\, \phi(c) = u]$, and is equal to the probability that $c$ was chosen by `rand_decr` at least $M$ times during these $u$ operations. This is the Binomial probability

$$\sum_{i=M}^{u} \binom{u}{k} \alpha^k (1-\alpha)^{u-k}$$

The probability that $c$ is not set during $u$ consecutive operations is $\Pr[\phi(c) = u] = (1 - l/m)^u$. Let $\delta = (1 - l/m)$.

Thus, the probability of $v[c] = 0$ after $U$ operations is $\sum_{u=M}^{U} \Pr[\phi(c) = u] \times \Pr[c = 0 | \phi(c) = u]$, or

$$\Pr[\text{FN}] = \sum_{u=M}^{U} \delta^u \sum_{k=M}^{u} \binom{u}{k} \alpha^k (1-\alpha)^{u-k} \qquad (1)$$

This probability is hard to express in closed form, since even the partial Binomial probability in the inner sum has no known closed form except in terms of the incomplete Beta function. Instead, we will attempt to bound this probability from above. We note that $\alpha = t/m$ is very small since $t \ll m$, and we can use the Poisson approximation to the Binomial distribution. That is, we can write $\binom{n}{k}\alpha^k(1-\alpha)^{n-k} \approx \frac{(n\alpha)^k}{k!}e^{n\alpha}$. Using this approximation, we rewrite Equation 1 as

$$\begin{aligned}
\Pr[\text{FN}] &= \sum_{u=M}^{U} \delta^u \sum_{k=M}^{u} \frac{(u\alpha)^k}{k!} e^{u\alpha} \\
&= \sum_{u=M}^{U} (\delta e^{\alpha})^u \sum_{k=M}^{u} \frac{(u\alpha)^k}{k!} \\
&= \sum_{u=M}^{U} (\delta e^{\alpha})^u \frac{(u\alpha)^M}{M!} \sum_{k=1}^{u-M} \frac{(u\alpha)^k}{k!}
\end{aligned}$$

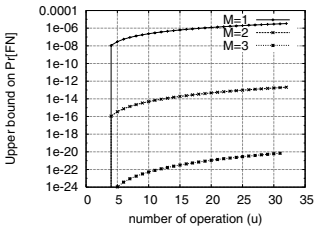



Fig. 10. Probability of guessing false negatives for $1 \leq u \leq 31$

Clearly, $\sum_{k=1}^{u-M} \frac{(u\alpha)^k}{k!} < e^{u\alpha}$, so that

$$\begin{aligned}
\Pr[\text{FN}] &< \sum_{u=M}^{U} (\delta e^{\alpha})^u \frac{(u\alpha)^M}{M!} e^{u\alpha} \\
&= \frac{\alpha^M}{M!} \sum_{u=M}^{U} u^M (\delta e^{2\alpha})^u \\
&< \frac{\alpha^M}{M!} U^M \sum_{u=M}^{U} (\delta e^{2\alpha})^u \\
&= \frac{(\alpha U)^M}{M!} (\delta e^{2\alpha})^M \frac{1-(\delta e^{2\alpha})^{(U-M-1)}}{1-(\delta e^{2\alpha})}
\end{aligned}$$

If we write $b = \delta e^{2\alpha}$, we have

$$\Pr[\text{FN}] < \frac{(\alpha U b)^M}{M!} \cdot \frac{1-b^{(U-M-1)}}{1-b} \tag{2}$$

Figure 10 shows how this upper bound on the probability of false negatives changes with $M$. As in [13], we set $t = 3$, $l = 4$ and $m = 11073741824$. The three curves correspond to cases when $M = 1$, $M = 2$ and $M = 3$. Each curve presents the change in the probability as $U$ varies from 1 to 31.

### D. Overall Success Probability for a Dishonest Broker

A replayed report will cause a false negative in the Bloom filter with probability $\Pr[\text{FN}]$, so that the publisher considers this as a fresh report with this probability. For replays, the broker can provide a valid public key certificate on demand, so the broker succeeds with probability $\Pr[\text{FN}]$ for replays.

For fabricated reports, the broker will be challenged with probability $q$ for a public-key certificate, so that he goes scot free with probability $1 - q$. Let the fraction of clients who choose to remain anonymous be $\psi$. The broker can decline to provide the public-key certificate, without arousing suspicion, for a fraction of challenges no higher than $\psi$. So the expected probability of success on the challenges is $q\psi$. The overall probability of success here is $1 - q + q\psi = 1 - (1 - \psi)q$.

If a dishonest broker replays reports with probability $p_r$ and fabricate reports with probability $p_f$, his overall success probability is $(\Pr[\text{FN}] \cdot p_r) + ((1-(1-\psi)q) \cdot p_f)$. A publisher can now choose $q$ and the SBF parameters to set this success probability at any level he deems appropriate.

**Claim VIII.1.** *Brokers are never penalized if they forward genuine reports to publishers.*

*Proof:* Let signature $sig_s(I_d|p)$ corresponding to a valid access arrive at publisher $p$. If $sig_s(I_d|p)$ causes a hit in SBF, $p$ will run the challenge-response protocol to verify that $sig_s(I_d|p)$ is a false positive. If $sig_s(I_d|p)$ does not causes a hit in SBF, $p$ considers it a non-replay report, with probability $q$ checks whether it is a fabricated report. Since the report is legitimate, this validation succeeds. Thus, no broker is penalized for forwarding a genuine report. □

## IX. Challenge and Response-based Solution for Publisher-payee Model

The broker can report undercounts in this model. The simplest solution is to have a subset of clients (the *reporters*) notify the publisher of each request they make. The broker does not know who the reporters are. If he is failing to report to the publisher a fraction $f$ of all client accesses, he will also fail to report fraction $f$ of the reporter accesses. Standard results from sampling theory [21] tell us that we can estimate $f$ accurately with a modest number of reporters (1000–2000), regardless of the client population size.

### A. Challenge-Response Approach to Notify Publishers

After receiving requested data from the broker, the reporter sends a report in the form of a *challenge* $c$ to the publisher, who verifies the challenge and returns a *response* $\rho$, which confirms that the report was received by the publisher.

Optionally, the publisher may solicit additional reporters by rewarding the them with e-cash sent with $\rho$. To stop broker tampering or e-cash interception and replay, the publisher signs a hash of the e-cash, concatenated with the challenge-response pair. Techniques in [11], [9] ensure that e-cash cannot be reused. Such schemes can also trace the publisher to whom the e-cash was issued.

### B. The Challenge-Response-based Protocol

Our protocol is presented in Figure 11. The sequence of message exchanged in the protocol is shown in Figure 12. To directly address the issue of broker tampering, *we pretend in these protocols that the broker explicitly forwards all messages between the reporter and the publisher*. We then show that the protocol remains tamper-proof.

### C. Properties of the Ciphertext Challenge

In Figure 11, the ciphertexts $c_{I_d}$ notify the publisher of the delivery of data item $I_d$ to a reporter. We identify the following three requirements for our cryptosystem.

A) Label Attachability: The encryption function $E()$ should allow the reporter to attach the identity of data item $I_d$ (the *label*) to the ciphertext. Labeling has been previously used in [5], [10].
B) Label Verifiability: The decryption process should allow the publisher to verify whether the reporter had attached the given label to the ciphertext. $D()$ and $V()$ denotes the decryption and the verification function, respectively.
C) Label Non-malleablity: It should be computationally infeasible to remove the label attached by the reporter.

Encryption functions such as those in [10][12] that are secure against adaptive-chosen ciphertext (CCA-2) attack [20] satisfy these requirements, and can be used to construct $c_{I_d}$.

```
Access(s, B, σ) :
  A1-A4) Execute steps A1 to A4 from the access
    protocol defined in Figure 3.
  A5) s executes
      M_r = Rand_string();
      c_{I_d} = E(M_r, π_p, I_d|p);
  A6) s sends {c_{I_d}, I_d|p} to b_m
  A7) b_m forwards {c_{I_d}, I_d|p} to p
  A8) If V(c_{I_d}, I_d|p) = false then p executes
      Penalty(b_m); HALT
  A9) If V(c_{I_d}, I_d|p) = true then p executes
      ++count_p;
      ρ = D(c_{I_d});
    9.1) p computes sig_p(e|c_{I_d}|ρ), where e is the
     e-cash reward.
    9.2) p sends {e, sig_p(e|c_{I_d}|ρ), ρ} to s; HALT
  A10) If ρ = M_r and e, sig_p(e|c_{I_d}|ρ) are valid
     then s executes HALT
       else s executes
       Penalty(b_m); HALT
```

Fig. 11. The Challenge-Response-based Protocol

**Claim IX.1.** *Undercounts do not occur in the protocol when the reporters are honest.*

*Proof:* In the $Access$ protocol, let reporter $s$ send message $\{c_{I_d} I_d|p\}$ in Step A5 of Figure 11. There are two possible ways for $b_m$ to avoid reporting the access to $p$, and yet forward a correct response to the reporter.

The first option is for $b_m$ to create another ciphertext $c'_{I_d}$, such that $D(c'_{I_d}) = D(c_{I_d})$ so that the verification of $\{c'_{I_d}, I'_d|p'\}$ at the publisher $p$ returns *true*, where $I'_d$ is a valid data identifier and $p'$ is some valid publisher. However, as $E()$ is CCA-2 secure, it is computationally infeasible for $b_m$ to construct such a $c'_{I_d}$ and then compute the response.

The second option is for $b_m$ can try to decrypt $c_{I_d}$. This is also not possible, because $E()$ is CCA-2 secure.

## X. Client-Broker Collusion

We now consider the scenario where a dishonest broker colludes with some clients. We expect most clients to be honest, so only a small fraction of dishonest clients collude in cheating the publisher.

**Definition 6.** *A client is considered* **dishonest** *if it participates in the generation of a fake report or if it helps the broker in hiding the genuine reports from the publisher.*

### A. Identifying Dishonest Clients in Broker-payee Model

We will require the broker to include the public key $\pi_s$ of the client in each report. In the absence of a certificate or some other binding to identity, $\pi_s$ itself contains no information about the client's identity. Consequently, $\pi_s$ appears as a random tag to the publisher, who gains no information about the identity of $s$.

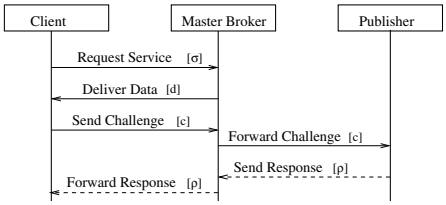



Fig. 12. Sequence Diagram for Challenge-Response-based Protocol

Now, $p$ uses a hash table to maintain a count for number of validated reports that included each public key. As we have already argued, the broker gains a premium for dishonesty in proportion to the overcount. For a large profit, this overcount must be large. If the number of dishonest clients is small, this large overcount will cause the entries for these clients to be disproportionately large. When the publisher detects this sort of anomaly, it requests explicit verification of identity, or takes other measures, as determined by policy considerations.

### B. Tolerating Collusions in Publisher-payee Model

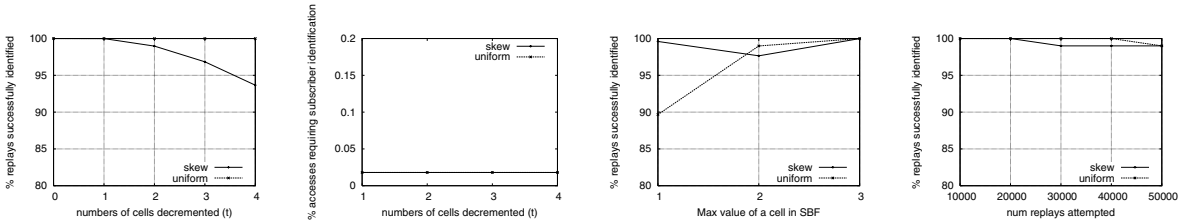
Consider a scenario where a legitimate data access caused the delivery of data item $d$. The report corresponding to this access goes *unaccounted* if the broker does not notify the publisher about the data access.

Our techniques cannot completely stop client-broker collusions. Hence, we propose that our protocol be used in scenarios where a single unaccounted report results in very small profit for the dishonest broker, for example MP3 song downloads for Apple's IPod. So, to generate a huge profit, the broker needs to hide a large number of genuine reports. Under the assumption that only a small number of clients are likely to collude, each dishonest client would be required to participate in hiding a large number of genuine reports. This is unlikely because the client would have to pay for a large number of data accesses, without any benefits for doing so. Hence, we say that the publisher tolerate client-broker collusion that could generate small frauds, but need not worry about collusions that may result in huge losses.

## XI. Experimental Evaluation

We used the MSNBC anonymous web data [2] as a real-world dataset to test our techniques. This dataset characterizes the pages visited by users who visited the MSNBC website for one day. URLs for MSNBC website categories, such as "frontpage", "news" or "tech" are considered as individual items in our experiments. The webpages for the various categories are accessed on an average 5.7 times by a total of 989,818 clients. We augmented each of these 4,698,795 accesses with a 1024-bit signature to create verifiable reports.

A cheating pattern is *skewed* if the broker replays some accesses after forwarding all the genuine accesses. It is *uniform* if the broker distributes the replays uniformly over the one-day period. The publisher maintains a SBF with the following




(a) Effect of varying $t$.

(b) Accesses requiring client ID, $0 \leq t \leq 4$

(c) Effects of varying $M$.

(d) Replays caught

Fig. 13. Simulation Results for Broker-payee Model

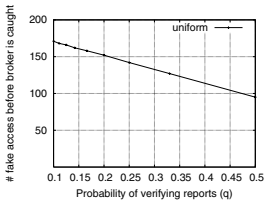



Fig. 14. Probability of Fake Reports Going Undetected is Low

default parameters: SBF vector size 8.9MB (75,180,720 bits) using two hash functions, with two bits decremented per execution, with each cell holding a max value of 3.

Figure 13 shows our simulation results for the SBF-based protocol. Figure 13(a) shows that our protocol very efficiently identifies replays, with a success rate of 99.99%. The success rate drops, as expected, to 93% as we increase the number of cells decremented at each operation to 4. We recommend decrementing no more than two cells at each operation.

In Figure 13(b) shows that the fraction of clients who need to give up anonymity is negligible (at 0.018%), which is the number of false positives in our protocol. Figure 13(c) shows that the rate of FNs, and hence the % replays identified, can be maximized by setting the maximum value in each cell to 3. Finally, Figure 13(d) show that our protocol is scalable and consistently identifies dishonesty at a high success rate.

Figure 14 shows that the number of fake accesses that go undetected before getting caught is very low, assuming that 5% of the total 4,698,795 reports are fraudulent (the broker wants to generate 5% profit). Half the fraudulent reports are replays and the other half are fabricated, and 5% of the clients are willing to be identified. The x-axis represents $q$, the probability with which the publisher verifies any broker report.

## XII. Conclusion

We have presented a SBF-based protocol that stops replay and fabrication of data accesses in a broker-payee model. This solution efficiently identifies both kinds of fake accesses and flags clients that collude with the broker to fool the publisher. Our solution for publisher-payee ensures that unaccounted data accesses cannot go unreported, unless clients collude with a broker. This solution tolerates a small degree of cheating and hence is applicable in payments schemes that attach low monetary value per data access.

**Acknowledgments:** This project was supported by a grant from Tata Consultancy Services, Inc.

## References

[1] Google click fraud, http://www.law.com/jsp/article.jsp?id=1153213525657.
[2] Msnbc dataset, http://kdd.ics.uci.edu/databases/msnbc/msnbc.html.
[3] B. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. *Lecture Notes in Computer Science*, 2045, 2001.
[4] V. Anupam, A. Mayer, K. Nissim, B. Pinkas, and M. K. Reiter. On the security of pay-per-click and other Web advertising schemes. *Computer Networks (Amsterdam, Netherlands: 1999)*.
[5] N. Asokan and V. Shoup. Optimistic fair exchange of digital signatures. *EUROCRYPT '98*, 1998.
[6] M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, and M. Waidner. iKP – A family of secure electronic payment protocols. pages 89–106.
[7] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
[8] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229, London, UK, 2001. Springer-Verlag.
[9] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. *EUROCRYPT'05, volume 3494 of LNCS, pp. 302-321*.
[10] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. *Crypto 2003*.
[11] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash ). *CRYPTO '88, pp. 319-327.*, 1989.
[12] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. *LNCS*, 1462, 1998.
[13] F. Deng and D. Rafiei. Approximately detecting duplicates for streaming data using stable bloom filters. In *ACM SIGMOD'06*.
[14] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto '86*.
[15] L. Fiege, A. Zeidler, A. P. Buchmann, R. Kilian-Kehr, and G. Mühl. Security aspects in publish/subscribe systems. In *Third Intl. Workshop on Distributed Event-based Systems'04*.
[16] M. K. Franklin and M. Yung. Secure and efficient off-line digital money. In *ICALP '93*, pages 265–276, London, UK. Springer-Verlag.
[17] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17, 1988.
[18] A. Metwally, D. Agrawal, and A. E. Abbadi. Duplicate detection in click streams. In *WWW '05*, pages 12–21.
[19] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13, 2000.
[20] C. Rackoff and D. Simon. Non-interactive zero-knowledge proofs of knowledge and chosen-ciphertext attack. *LNCS, CRYPTO 91*.
[21] H. Stark and J. W. Woods. *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice Hall, USA, 1994.
[22] D. R. Stinson. Cryptography: theory and practice. *CRC Press*, 1995.
[23] A. Tanner and M. A. Jaeger. Pricing in publish/subscribe systems. In *ICEC '04*.
[24] C. Wang, A. Carzaniga, D. Evans, and A. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. In *HICSS'02*.