A Hybrid Method to Detect Deflation Fraud in Cost-Per-Action Online Advertising

Xuhua Ding

Singapore Management University
Email: xhding@smu.edu.sg

Abstract. Web advertisers prefer the cost-per-action (CPA) advertisement model whereby an advertiser pays a web publisher according to the actual amount of transactions, rather than the volume of advertisement clicks. The main obstacle for a wide deployment of this model is the deflation fraud. Namely, a dishonest advertiser under-reports the transaction count in order to discharge less. In this paper, we present a mechanism to detect such a fraud using a hybrid of cryptography and probability tools. With the assistance from a small number of users, the publisher can detect deflation fraud with a success probability growing exponentially with the fraud amount, and can estimate the amount of frauds. Our scheme is amiable to both the advertiser and the users because the existing transaction model remains unchanged. It is also efficient and scalable as the incurred communication, computation and storage costs are independent of the number of transactions.

1 Introduction

Cost-per-action (CPA) is gathering its popularity among online advertisers due to its cost-effectiveness. Different from the cost-per-click model where an advertiser pays the web publisher for every user click, the payment in the CPA model is based on the amount of predefined user action, e.g. downloading, sale or sign-up. Nonetheless, such a model is not favored by the web publishers, because a dishonest advertiser may undercount the actions and consequently pays less commission fee. This type of cheating is called *deflation fraud*. Such frauds can also be found in other applications. For example, in publish-subscribe networks [7, 14], an event publisher shares profits with the brokers, and a dishonest broker can undercount the number of subscriptions. In online content distribution businesses, a content distributor may cheat a content provider in a similar fashion.

A dual problem of deflation fraud is *inflation fraud*, whereby an entity cheats by maliciously over-counting or over-reporting transactions for its financial advantages. Although there exist many schemes proposed to address the inflation fraud as in [8, 17, 18, 15, 9, 11, 20], the deflation fraud has not caught sufficient attention except in [10]. As explained in [10], the philosophy for all inflation fraud detection schemes is knowledge proof. Intuitively, an inflation adversary is challenged to prove sufficient knowledge pertaining to the inflated count. Unfortunately, the same tactic fails for deflation fraud detection, as no scheme can challenge an adversary to present a proof on *absence of knowledge*. Therefore, the general approach for deflation detection is that the verifier (e.g. the web publisher in our context) gathers as much information as possible regarding the claim made by the prover (e.g. the advertiser). The solutions used in [10] are based on an online trusted third party (TTP) which mediates the transactions. A similar approach taken by Google's AdWords is that the verifier watches over the transactions directly. Obviously, the more information the verifier gathers, the stronger the detection scheme is.

Obviously, the aforementioned approaches are inefficient and unscalable as they are intrusive to the advertiser's business operation and involve an online TTP. We observe that missing a few transactions is a tolerable counting error to Ps as long as the fraction of fraud is sufficiently small. Therefore, we design an efficient and flexible scheme by slightly relaxing the security. The main results of this paper is a cryptography and probability based deflation-fraud detection scheme with the following attractive features.

- It detects any z amount of deflation frauds with a success probability at least growing exponentially with z. The web publisher can tune a security parameter to strike a balance between a high security assurance and a low cost.
- It allows the web publisher to estimate the expected transaction amount, which provides a sound basis to detect any frauds in a large magnitude.
- It is not intrusive to the advertiser in the sense that the transactions only involve the advertiser and its users only. It is also user-friendly as end-users do not need to maintain any secret information.
- It is efficient and scalable. The communication, computation and storage costs incurred by our scheme are independent of the amount of transactions.

The rest of the paper is organized as follows. We discuss related work in Section 2 and the building blocks in Section 3. Then, we formulate the problem in Section 4. Section 5 proposes the deflation fraud detection scheme. We analyze its security and performance in Section 6, then conclude the paper in Section 7.

2 Related Work

The most relevant work is by Johnson and Staddon [10]. They considered deflation fraud in content distribution. Three schemes are proposed in [10]. In the first scheme, the verifier impersonates regular users using different pseudonyms. Then, it checks whether its pseudonyms appears in the content distributor's report. The second scheme involves a TTP to pre-issue a set of keys to users in certain distribution. The content distributor must use a key known to all users, which helps the auditor to estimate the user set size. The third scheme is similar to the second one with the difference being a reduced user storage. The constructions of [10] has obvious architecture drawbacks and are not applicable to online advertising.

The problem investigated in this paper is akin to the count integrity in publish-subscribe networks [7, 14]. A publish-subscribe system [7] involves a publisher which is the data source, a set of subscribers who receive their preferred data items, and a broker network consisting of a set of broker nodes which disseminate data from the publisher to the subscribers. The brokers report to the publisher about the amount of data delivered to the subscribers. The approach to count integrity proposed in [14] requires the publisher to participate in *all* data delivery transactions. This approach obviously abandons the design principle of publish-subscribe systems whose primary goal is to decouple publishers and subscribers in order to be more scalable and to save the publisher from the heavy workload of data delivery.

Web metering mainly deals with *overcount fraud* (a.k.a. *inflation attacks*). Naor and Pinkas [17] proposed a secret sharing based scheme to verify the number of users served by a web server. Their scheme is not suitable for generic online transactions since it requires the audit agency to initialize every user before running transactions. A special form of inflation attack is the well-known *click fraud*, whereby the adversary cheats on the amount of website visits instead of transactions. Gandhi et. al proposed in [9] countermeasures based on construction of advertisement code. Similar works also include [11, 20, 18, 15, 9].

In [12], Markus Kuhn proposed a novel approach to probabilistically counting a large collection of digital signatures, which may be used in applications like web page metering or ranking mechanisms. Though very efficient, this scheme suffers from inaccuracy, since it only provides an estimation on magnitude. However, many business applications demand a more precise count. Moreover, it does not address deflation fraud.

Another line of research related to our study is electronic voting schemes, such as [6, 5, 16]. Among many security requirements such as receipt-freeness, a fundamental requirement is that the ballots should be tallied correctly. Although secure e-voting schemes can theoretically defeat inflation/deflation attacks, they are not suitable for online advertising or content distribution, mainly because e-voting has a special and expensive infrastructure and has a heavy toll on the computation/communication costs.

3 Building Block and Notations

The cryptographic building block used in our scheme is the signature of knowledge [4, 13], a non-interactive form of the zero-knowledge proof. The most primitive signature of knowledge is Schnorr signatures [19], whereby the signer proves that she knows the discrete logarithm of y to the base g in a cyclic group $G = \langle g \rangle$. An extension of Schnorr signatures can be used to prove the equality of two discrete logarithms. Suppose that g and g' are two generators of group G, and $y = g^x, y' = g'^x$. Knowing x, the signer produces a signature of knowledge proving that $DLOG(y, g) = DLOG(y', g')$. In essence, the signer generates a tuple which can be treated as two Schnorr signatures sharing the same challenge. We denote it by SKELOG$[x : y = g^x \wedge y' = g'^x]$.

In [4], Camenisch and Stadler defined the signature of knowledge of a *double discrete logarithm* of y to the base g and a. Let $y = g^{a^x}$. Knowing x, the signer computes $(c, s_1, \cdots, s_l)$ as the signature of knowledge denoted as SKLOGLOG$(x : y = g^{a^x})$, where l is a security related parameter. Their scheme is essentially a non-interactive version of l rounds of zero-knowledge proof with l binary challenges. Since the challenge used in each round is one bit, the signer can successfully cheat the verifier in one round with a probability $1/2$. Therefore, an adversary can forge SKLOGLOG$[x : y = g^{a^x}]$ with a probability $1/2^l$. When l is sufficiently large, the probability is negligible.

In this paper, we combine SKLOGLOG$[x : y = g^{a^x}]$ and SKELOG$[x : y = g^x \wedge y' = h^x]$, so that with the knowledge of x, the signer can produce a signature of knowledge proving that the double discrete logarithm of $y = g^{a^x}$ equals to the discrete logarithm of $y' = h^x$. We denote it as SKELOGLOG$[x : y = g^{a^x} \wedge y' = h^x]$. Note that the computation of a^x in computing y and the computation of h^x are in the same group. The details are shown in Section 5.

4 Problem Formulation

4.1 System Overview

A typical online advertisement system consists of three types of entities: a web publisher Ps offering the advertising service through its web or search engines; a (unknown-sized) set of users $\mathcal{U}$; and an advertiser Ad who offers services or products to users. Ps displays to users the advertisement for Ad. By clicking the advertisement, the user is redirected to Ad's web site. The user is said to perform a *transaction* if he signs up the service or downloads the product. The user may or may not perform a transaction, solely depending on his own willingness. Periodically, Ad reports to Ps the number of transactions contributed by Ps's advertisement. In deflation fraud, Ad under-reports to Ps in order to pay less commission fee.

Our goal is to allow Ps to detect Ad's cheating in an efficient fashion. In a nutshell, the basic approach is to collect information from Ad and a tiny subset of $\mathcal{U}$. By analyzing the received data, Ps can discover (or suspect) the fraud in a probability growing with the fraud amount. The proposed scheme consists of the protocols/algorithms listed below.

- Initialization: Both Ps and Ad are initialized with the proper states. Ps authorizes Ad to run n transactions in maximum within one billing cycle.
- Advertising: In this protocol, a user $U \in \mathcal{U}$ interacts with Ps, where an advertisement of Ad is shown to U.
- Transaction: U may interact with Ad for an transaction, e.g. sign-up. It receives a receipt signed by Ad.
- Feedback: With a probability ρ, U runs this protocol with Ps to return his receipt, where ρ is a system-wide parameter selected by Ps. (The choice of ρ is discussed in Section 6.)

– Report: Ad reports to Ps about the amount of transactions performed. Ps detects fraud based on both the receipts collected from users and the report from Ad, and estimates the fraud amount (if any).

4.2 Assumptions, Adversary Model and Security Notions

We assume that every user $U \in \mathcal{U}$ is independent, and their protocol executions are regarded as independent events. We assume that all communication channels are confidential and authentic, e.g. via SSL/TLS connections. Henceforth, we do not consider attacks on the communication channels.

The adversary in our scheme is Ad. If there are in fact N transactions, Ad's objective is to report to Ps a fraudulent transaction count $N' < N$ without being detected. We assume that Ad is rational in the sense that it would not risk the exposure of its long term secret and it sets a risk threshold for itself. We do not consider collusion attacks between Ad and corrupted users. This is because Ad can always run the transaction with its colluders without executing the prescribed protocol[1].

The security strength of our scheme is defined based on the upper bound of the detection miss probability with regard to the amount of frauds.

Definition 1 (Deflation Resistance). *A deflation fraud detection scheme involving a publisher* Ps *and an advertiser* Ad *is said to be (z, ν)-secure for $\nu \in [0, 1]$ and $z \in \mathbb{N}$, if and only if the probability that* Ad *successfully undercounts $x \geq z$ transactions without being detected is bounded by ν, i.e.* $\mathsf{Pr}[\textit{detection fail } |x \geq \delta] \leq \nu$.

A perfect detection scheme should be $(1, 0)$-secure. Namely, Ps successfully detects any amount of deflation fraud. This can only be achieved by supplying Ps with the complete information about Ad's transactions. The aforementioned naive approaches, for instance, by introducing an online TTP, fall in this category. Our goal is to construct an efficient and scalable scheme by relaxing the security strength slightly, as long as z and ν are small enough to meet the application needs.

5 The Scheme

A high level view of the proposed scheme is as follows. Ps delivers a sequence of hash tokens to Ad. When Ad performs a transaction, a receipt derived using a fresh token is returned to the user. A small set of users report their receipts to Ps. The latter verifies whether Ad honestly runs the prescribed protocol and detects any anomaly using the received data. The details are presented below. In the sequel, we use the notations listed in Table 1.

[1] Note that cryptography techniques alone can not detect collusion attacks. Promising solutions could be based on trusted hardware or TTP. Nonetheless, we remark that it is infeasible for Ad to collude with a large portion of Internet users.

Notations	Description
(x, y)	Ad's private and public key pair for signing receipts to users;
(u, v)	Ad's another private and public key pair for token usage proof;
t_i	the i-th tokens issued to Ad;
C	Ad's transaction counter;
I_{click}	Ps's click counter;
C_f	the maximum transaction count received by Ps.
I_f	the click count corresponding to C_f.
N	the actual amount of transactions;
M	the total number of advertisement clicks;
ρ	the probability for a user to run Feedback after Transaction.

Table 1. Table of Notations

5.1 Initialization

Given a system wide security parameter κ, all participants agree on the following cryptographic setting. Let p, q, q' be three large primes satisfying $q|p-1$ and $q = 2q' + 1$, and the discrete logarithm problem is intractable in both $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$. Let $g \in_R \mathbb{Z}_p^*$ such that $\mathbb{G} = \langle g \rangle$ is a cyclic subgroup of Z_p^* of order q. Let $h \in_R \mathbb{Z}_q^*$ such that $\mathbb{G}_h = \langle h \rangle$ is a cyclic subgroup of Z_q^* of order q'. Hereafter, we omit the modulus p and q for group operations in $\mathbb{G}$ and $\mathbb{G}_h$ respectively, if they are indicated from the bases in use. Let $H : \mathbb{Z}_q \rightarrow \mathbb{Z}_q$ be a collision resistant hash function; and let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^k$ be a collision resistant hash function, where k is a parameter determined by κ, e.g. $k = 160$. Let l be a parameter for SKELOGLOG determined by κ as well, e.g. $l = 80$.

Ps authorizes Ad to generate a chain of n hash tokens for n transactions. The benefits of using a hash chains are: 1) to save the communication and storage cost (i.e. a seed can generate the entire chain.); 2) to model the hash function as a random oracle for the provable security, because a hash token will be used in Transaction to derive a random number for signing a receipt. Note that the communication cost for the protocol is constant. The protocol of initialization is shown in Figure 1.

5.2 Advertising

Ps displays Ad's advertisements, e.g. a flash or a banner, on its web pages. It also maintains a counter I_{click} to keep track of the number of advertisement clicks. Initially, $I_{click} = 0$. Advertising begins when U clicks the advertisement. Then, Ps sends the (I_{click}, t_0) to U. Then, Ps sets $I_{click} \leftarrow I_{click} + 1$. To avoid confusion, we use M to denote the final I_{click}, i.e the total number of advertisement clicks.

Initialization Protocol (by Ps and Ad)

1. Ps executes the following:
   (a) Generate a random seed $T \in_R \mathbb{Z}_q$; choose $n \in \mathbb{N}$ as the maximum amount of transactions Ad can perform;
   (b) Set $t_0 \leftarrow H^n(T)$, where $H^n(T) \triangleq \underbrace{H(\dots H}_{\text{n times}}(T)\cdots)$;
   (c) Send $\{T, t_0, n\}$ to Ad;
2. Ad executes the following:
   (a) Select $x \in_R \mathbb{Z}_q$ and set $y \leftarrow g^x \bmod p$.
   (b) Select $u \in_R \mathbb{Z}_{q'}$ and set $v \leftarrow h^u \bmod q$
   (c) Output $PK := (y, v, p, q)$ as Ad's public key and $SK := (x, u)$ as its private keys. Ad's receipt signature key pair are (y, x).
   (d) On receiving $\{T, t_0, n\}$, check whether $t_0 = H^n(T)$. Abort if not equal. Otherwise, accept them.

Fig. 1. The Initialization protocol

5.3 Transaction

Ad maintains a counter denoted by C to count the number of transactions. Initially, $C = 0$. After running Advertising, U may decide to run a transaction with Ad. Similarly, a user can also decide not to run Transaction.

In the protocol, U first sends its I_{click} to Ad requesting for a transaction. Ad then signs I_{click} using its receipt signature key x together with a random number derived from the token t_C. As a result, Ad responds to U with two parts: a receipt which is a Schnorr signature on I_{click}, and a SKELOGLOG proof proving that the randomness in the Schnorr signature is derived from a hash token in the authorized hash chain. U completes the transaction if both the signature and the proof are verified true. The protocol details are described below in Figure 2.

5.4 Feedback

After running Transaction, U may choose to run Feedback to send a feedback to Ps. Let ρ denote the expected probability for a user to run Feedback after Transaction[2]. In the protocol, U simply returns its I_{click} and its receipt (C, t_C, γ, β) to Ps. Let Λ denote all receipts received. Let C_f denote the largest C in Λ, and I_f denote the I_{click} paired with C_f. When a new receipt is inserted into Λ, Ps runs the function Fraud(Λ) which returns 1 if there exist two distinct receipts with the same hash token. The details are described in Figure 3. Initially, $\Lambda = \emptyset$ and $C_f = 0$.

[2] Ps sets ρ by using financial tools, e.g. receipt redemption or lucky draw, to attract users to run the protocol.

Transaction Protocol (by Ad and U)

Ad's input : $\{T, x, u\}$, the present transaction counter C; U's input: $\{I_{click}, t_0\}$.

1. Ad executes the following steps, when receiving I_{click} from U.
   (a) Compute $t_C \leftarrow H^{n-C}(T)$, and $\alpha \leftarrow \mathcal{H}(t_C \| \mathsf{Ad})$.
   (b) Compute $r \leftarrow \alpha^u \bmod q$.
   (c) Compute a Schnorr signature (γ, β) as $\gamma = \mathcal{H}(t_C \| y \| g \| I_{click} \| g^r)$ and $\beta = r - x \cdot \gamma \bmod q$.
   (d) Randomly selecte $r_1, \cdots, r_l \in_R \mathbb{Z}_{q'}$
   (e) Compute $w_i \leftarrow h^{r_i}$ and $w'_i \leftarrow g^{(\alpha^{r_i})}$, for $i = 1, \cdots, l$.
   (f) Set $\psi \leftarrow \mathcal{H}(v \| h \| g^r \| g \| t_C \| w_1 \| \cdots \| w_l \| w'_1 \| \cdots \| w'_l)$.
   (g) For $i = 1, \cdots, l$, set

$$\lambda_i = \begin{cases} r_i & \text{if } \psi[i] = 0, \\ r_i - u \bmod q' & \text{otherwise.} \end{cases}$$

   (h) Send $\{C, t_C, (\gamma, \beta), (\psi, \lambda_1, \cdots, \lambda_l)\}$ to U; then $C \leftarrow C + 1$.
2. $\mathcal{U}$ performs the following on receiving $\{C, t_C, (\gamma, \beta), (\psi, \lambda_1, \cdots, \lambda_l)\}$,
   (a) Verify whether t_C is in the hash chain rooted at t_0. Namely, if $t_0 \neq H^C(t_C)$, return 0 and abort;
   (b) Compute $\alpha \leftarrow \mathcal{H}(t_C \| \mathsf{Ad})$.
   (c) Evaluate the transaction verification function denoted by $V(t_C, I_{click}, \gamma, \beta, \psi, \lambda_1, \cdots, \lambda_l)$ as follows.
      i. Compute $W \leftarrow g^\beta y^\gamma$.
      ii. If $\gamma \neq \mathcal{H}(t_C \| y \| g \| I_{click} \| W)$, return 0 and abort;
      iii. For $i = 1, \cdots, l$, compute

$$w_i = \begin{cases} h^{\lambda_i} & \text{if } \psi[i] = 0, \\ v^{\lambda_i} & \text{otherwise.} \end{cases} \text{ and } w'_i = \begin{cases} g^{(\alpha^{\lambda_i})} & \text{if } \psi[i] = 0, \\ W^{(\alpha^{\lambda_i})} & \text{otherwise.} \end{cases}$$

      iv. If $\psi \neq \mathcal{H}(v \| h \| W \| g \| t_C \| w_1 \| \cdots \| w_l \| w'_1 \| \cdots \| w'_l)$, return 0 and abort.
   (d) Accept (C, t_C, γ, β) as a receipt for this transaction and return 1.

Fig. 2. The Transaction Protocol

Note that the first step in $\mathsf{Fraud}(\Lambda)$ does not require additional modular exponentiations since $g^\beta y^\gamma$ has been computed when verifying the Schnorr signature (γ, β). To save the time cost for finding cheating, Ps can make use of a Bloom Filter to test the membership of $g^\beta y^\gamma$.

5.5 Report

Report is run by Ad and Ps at the end of each billing cycle. In the protocol, Ad reports to Ps with $\hat{C}$ as the number of transactions. In a deflation fraud, $\hat{C}$ is less than the actual number of transactions (denoted by N). Since $\mathsf{Feedback}$

```
Feedback Protocol (by Ps and $U$)
Ps's input: $\Lambda, C_f$; $U$'s input: $\{C, t_C, I_{click}, \gamma, \beta, \psi, \lambda_1, \cdots, \lambda_l\}$.

 1. $U$ sends to Ps: $\{I_{click}, C, t_C, \gamma, \beta, \psi, \lambda_1, \cdots, \lambda_l\}$.
 2. Ps executes the following steps.
    (a) Verify the receipt using the same algorithm in Figure 2. Abort if
        $V(t_C, I_{click}, \gamma, \beta, \psi, \lambda_1, \cdots, \lambda_l) = 0$.
    (b) Set $\Lambda \leftarrow \Lambda \cup \{(C, I_{click}, \gamma, \beta)\}$.
    (c) Execute Fraud($\Lambda$) as follows:
        i. If $\exists(C, I'_{click}, \gamma', \beta') \in \Lambda$, s.t 1) $I'_{click} \neq I_{click}$ and 2) $g^{\beta}y^{\gamma} = g^{\beta'}y^{\gamma'}$,
           compute $x \leftarrow (\beta-\beta')/(\gamma'-\gamma)$. Claim Ad's deflation fraud and return
           1.
       ii. If $C > C_f$, set $C_f \leftarrow C$, and $I_f \leftarrow I_{click}$. Return 0.
```

Fig. 3. The Feedback Protocol

is independent of the Report, the execution of Report implies that Ps does not discover cheating from Feedback. Therefore, Ps assesses the credibility of $\hat{C}$ using the data it receives.

The data includes M, C_f, I_f, where M is the total number of advertisement clicks and is the result from Advertising; C_f and I_f are from Feedback. With these data and ρ, Ps runs the following steps.

1. If $C_f > \hat{C}$, then Ps claims that Ad cheats and the amount of deflation fraud is at least $C_f - \hat{C}$.
2. Ps may runs an anomaly detection using $\hat{C}, \rho, C_f, I_f, M$. In short, Ps first computes $\mathbf{E}[N]$. An alarm will be raised if the difference between $\mathbf{E}[N]$ and $\hat{C}$ is larger than a positive threshold selected by Ps. Furthermore, Ps evaluates the probability that Ad cheats. If both are significantly large, Ps can seek for the intervene of a trusted party for auditing. Note that anomaly detection produces false positives and false negatives. More details are explained in Section 6.2.
3. Ps and Ad settle the payment. If $\hat{C} = n$, they reset the entire protocol. Namely, Ps issues a new batch of hash tokens to Ad by running a new round of Initialization. Otherwise, Ps and Ad continue to use the present batch of tokens until n transactions are performed.

6 Analysis

Recall that we do not consider collusion attacks. Therefore in order to produce a valid receipt, Ad has to either honestly execute all Transaction with fresh tokens or to cheat by using duplicated tokens. For easiness of discussion, we refer to the first type of attack by *withholding attack* and the second type of attack by *duplication attack*. Let $P_{w,z}$ denote the maximum probability of detection failure when Ad runs withholding attacks for z transactions only; and $P_{d,z}$ denote the maximum probability of detection failure when Ad runs duplication attacks for z transactions only.

Caveat. The withholding attack actually does not benefit Ad in the long run, as it will not get new authorization tokens until the current batch of n tokens are used up. If Ad undercounts for the present, it has to inflate the count back in the future. Moreover, as shown later in Lemma 3, the duplication attack is more advantageous to Ad. Although our scheme has such deterrence, we still include the withholding attack into our analysis for the completeness of the discussion.

We now first proceed to analyze the success probability that Ps catches duplication attacks. We then analyze how Ps further detects fraud by finding anomaly. Finally, we analyze other security properties and performance.

6.1 Token Duplication Detection

To prove the security strength of the scheme, we first show that one hash token only results in one unique random number in Ad's Schnorr signature (γ, β).

Lemma 1. *Let* $\sigma = (t, M, \gamma, \beta, \psi, \lambda_1, \cdots, \lambda_l)$ *and* $\sigma' = (t', M', \gamma', \beta', \psi', \lambda'_1, \cdots, \lambda'_l,)$ *be the user receipts for two transactions. If* $t = t'$ *and* $V(\sigma) = V(\sigma') = 1$*, then* $\mathsf{Pr}(g^{\beta} y^{\gamma} \neq g^{\bar{\beta}} y^{\bar{\gamma}})$ *is negligible.*

Proof. The proof is trivial. We show that $\mathsf{Pr}(g^{\beta} y^{\gamma} \neq g^{\beta'} y^{\gamma'}) < 2\epsilon$, where ϵ denotes the error probability of the signature of knowledge scheme.

Let $W = g^{\beta} y^{\gamma}$ and $W' = g^{\beta'} y^{\gamma'}$. Since $V(\sigma) = V(\sigma') = 1$, we have

$$\mathsf{Pr}[\text{DLOGLOG}(W,g,t) = \text{DLOG}(v,u)] = \mathsf{Pr}[\text{DLOGLOG}(W',g,t') = \text{DLOG}(v,u)] = 1-\epsilon$$

according to the soundness definition of signature of knowledge. Since $t = t'$, we have $\mathsf{Pr}(W \neq W') < 1 - (1-\epsilon)^2 < 2\epsilon$. Therefore, if the signature of knowledge scheme is sufficiently sound, the probability that one hash token results in different randomness is negligible. $\square$

Thus, if two users have verified their receipts with the same hash token t_C in Transaction, the probability that Ad has used the same randomness g^r in generating two different Schnorr signatures is overwhelming. This serves as the basis for Ps to catch Ad's fraud. We summarize it in the following lemma.

Lemma 2. *If* $\rho < 1/3$*, then* Ad*'s duplication attack with* z *tokens can escape detection with the maximum probability being* $(1-\rho^2)^z$*. Namely,* $P_{d,z} = (1-\rho^2)^z$*.*

Proof. (sketch) For each of the z duplication attacks, Ad computes a Schnorr signature with a duplicated hash token which has been used in another signature. According to Lemma 1, Ad will be caught if any two users return the same hash token.

To cheat z times, Ad has two exclusive tactics. One is to use z distinct tokens with each being used exactly twice, i.e. reused exactly once. The other approach is that there exists at least one token which is used more than twice in total. The first tactic is more optimal for Ad than the second one. It can be proved by using an induction on z to compare the two probabilities of Ad's successful evasion. A rigorous proof is in Appendix A. The intuition is as follows. For the first tactic, only when both users receiving the same hash token return their receipts to Ps, can Ad be caught. In contrast, for the second one, it allows *polynomially* more combinations of feedbacks. Therefore, the first tactic maximizes the likelihood for Ad to evade detection.

To be conservative, we evaluate the scheme's resistance to Ad's best tactic, i.e. no hash token is used more than twice. Therefore, there are z pairs of Schnorr signatures which share a common hash token. Note that every user runs Feedback independently. Hence, the probability that two users with the same hash token return is ρ^2. As a result, Ad can escape after cheating z times with a probability $(1-\rho^2)^z$. $\square$

Next, we show that the duplication attack is more advantageous to Ad, as it allows Ad to escape fraud detection with a higher probability than the withholding the same amount of transactions.

Lemma 3. *For a deflation fraud with z transactions,* Ad *has a higher probability to escape detection by the duplication attack only than by the withholding attack only for z transactions. Namely, $P_{w,z} < P_{d,z}$.*

Proof. In proving Lemma 2, we have shown that $P_{d,z} = (1-\rho^2)^z$. The withholding attack can only be detected in Report, when there exists $(C, I_{click}, \gamma, \beta)$ in Λ s.t. $C > \hat{C}$. Thus, for withholding z transactions, Ad evades detection as long as none of those z users ignored by Ad runs Feedback with Ps. The probability of that event to occur is $\mathsf{Pr}_{w,z} = (1-\rho)^z$, which is less than $P_{d,z}$. $\square$

From the lemmas above, we show the security strength of the proposed scheme in the following theorem.

Theorem 1. *The proposed scheme is $(z, (1-\rho^2)^z)$-deflation-resistant for $\rho < 1/3$.*

Proof. Suppose that Ad intends to deflate the transaction count by z in total. Without loss of generality, suppose that Ad withholds z_1 transactions and duplications z_2 tokens, s.t. $z = z_1 + z_2$ and $z_1, z_2 \geq 0$. Ad evades detection when Ps fails to detect both attacks. Thus $\mathsf{Pr}[\text{detection fail}|z] = P_{w,z_1} \cdot P_{d,z_2}$. From Lemma 3, $\mathsf{Pr}[\text{detection fail}|z] < P_{d,z_1} \cdot P_{d,z_2} = P_{d,z} = (1-\rho^2)^z$.

Alternative, we can prove this by showing $\mathsf{Pr}[\text{detection fail}|z] = (1-\rho)^{z_1}(1-\rho^2)^{z_2} = (1-\rho)^z(1+\rho)^{z_2}$ which reaches its upper bound when $z_2 = z$. Thus, the propose scheme is $(z, (1-\rho^2)^z)$-deflation-resistant. $\square$

Remark 1. Our scheme deters deflation fraud by revealing Ad's private key, in a similar fashion as in offline detection of double-spending e-cash [3]. Suppose that there exit two distinct user receipts (β_1, γ_1) and (β_2, γ_2) using the same random number r, i.e. $\beta_1 = r - x \cdot \gamma_1 \bmod q$ and $\beta_2 = r - x \cdot \gamma_2 \bmod q$. Thus, it is straightforward to derive x by computing $(\beta_1 - \beta_2)(\gamma_2 - \gamma_1)^{-1} \bmod q$. We argue that the private key extraction is a deterrence to Ad's duplication attacks.

Remark 2. It is an interesting challenge to design an efficient and practical $(1,0)$-deflation-resistant deflation fraud detection scheme. The difficulty stems from the efficiency requirement. We observe that it seems infeasible to detect all frauds by using cryptographic techniques alone, unless *all* transactions data are known to Ps.

Remark 3. When $\rho \geq 1/3$, the probability to detect fraud is even higher, because Ps collects more information with a higher ρ. Nonetheless we do not have a close form formula to describe the success probability. Therefore, the constraint on ρ in the above theorem is not an advantage to the adversary. We also argue that a practical ρ is typically small due to Ps's expense constraints.

6.2 Anomaly Detection

Even though Ad's attack possibly evades the token-duplication detection in both Feedback and Report, Ps can also detect the fraud by using probability analysis. Different from the previous detection which relies on the non-repudiable evidences, the probability analysis only indicates the likelihood of Ad's cheating.

Detection using C_f. Recall that from Feedback execution, Ps concludes that there are at least C_f transactions. Based on C_f, Ps estimates the expected N using the following theorem.

Theorem 2. *Let N be the random variable denoting the total number of transactions. Let X be the random variable denoting the maximum transaction count received in Feedback, whose space is $[0, N]$. Then $\mathbf{E}[N] > \mathbf{E}(X) + (1-\rho)/\rho$.*

Proof. Since every user is independent in running Feedback, thus $\mathsf{Pr}[X = c|N = n] = \rho(1-\rho)^{n-c}$. Thus, we have $\mathbf{E}[X|N = n] = \sum_{i=0}^{n} i\rho(1-\rho)^{n-i} = n - (1-\rho)/\rho - (1-\rho)^{n+1}/\rho$. Therefore, we have

$$\begin{aligned}
\mathbf{E}[X] &= \mathbf{E}[\mathbf{E}[X|N = n]] = \sum_{n} \mathbf{E}[X|N = n]\mathsf{Pr}[N = n] \\
&= \sum_{n} (n - (1-\rho)/\rho - (1-\rho)^{n+1}/\rho)\mathsf{Pr}[N = n] \\
&= \mathbf{E}[N] - (1-\rho)/\rho \sum_{n} \mathsf{Pr}[N = n] - \frac{1}{\rho}\sum_{n}(1-\rho)^{n+1}\mathsf{Pr}[N = n] \\
&< \mathbf{E}[N] - (1-\rho)/\rho
\end{aligned}$$

Therefore, $\mathbf{E}[N] > \mathbf{E}(X) + (1-\rho)/\rho$ which completes the proof. $\square$

In fact, if N is known to be in the range $(N_0, +\infty)$ where $(1-\rho)^{N_0+1} \approx 0$, Ps can even conclude that $\mathbf{E}[N] \approx \mathbf{E}[X] + (1-\rho)/\rho$. Next, we analyze how to estimate $\mathbf{E}[X]$ from the known information.

Let ϵ_A be Ad's risk threshold. Let Z be the total amount of duplication attacks Ad feels safe to perform in order to evade the token-duplication detection. Namely $(1-\rho^2)^Z < \epsilon_A$. To maximize its deflation fraud, Ad attempts to minimize

C_f as much as possible. Since each user runs Feedback independently, Ad cannot predict which user would return a receipt. We remark that for every transaction, Ad's cheating is prior to the user's Feedback. Thus, Ad's Z cheating is equivalent to reduce the transaction count by Z, since Z tokens are duplicated. Thus, the actual $\mathbf{E}[X]$ is only $\mathbf{E}[N] - Z - (1-\rho)/\rho$.

As Ps knows ρ, it can estimate $Z < z_0$ where $(1-\rho^2)^{z_0} \approx \epsilon_A$. Then, it estimates the expected N as $C_f + z_0 + (1-\rho)/\rho$ according to the above theorem. If the difference between the calculated $\mathbf{E}[N]$ and $\hat{C}$ is much larger than a predetermined threshold, Ps suspects that Ad has deflation frauds.

Detection using I_f. From the protocol execution, Ps observes that there exist $M - I_f$ users whose click counts are larger than I_f and none of them offers feedback. Let $d = M - I_f$ and let Ω denote this set of d users. According to Ad's report, there exist $k = \hat{C} - C_f$ transactions generated by Ω. Ps checks whether Ad cheats by executing the following steps.

1. Ps computes $\xi = C_f/I_f$ as the expected probability for one user making a transaction following an advertisement click.
2. Let P_i denote the probability of the event that there are i users in Ω who have run Transaction with Ad. Ps computes $P_i = \binom{d}{i}\xi^i(1-\xi)^{d-i}$.
3. Let a random variable A denote the number of feedbacks from Ω, and a random variable B denote the number of transactions from Ω with no feedback being sent to Ps. Let χ denote $\Pr[B > k | A = 0]$. Essentially, χ is the conditional probability that Ad has under-reported in the present circumstance. Ps evaluates χ as

$$\chi = \frac{\sum_{i=k+1}^{d} P_i(1-\rho)^i}{\sum_{i=0}^{d} P_i(1-\rho)^i} = \frac{\sum_{i=k+1}^{d} \binom{d}{i}\xi^i(1-\xi)^{d-i}(1-\rho)^i}{(1-\xi\rho)^d}$$

If χ is larger than a positive threshold, e.g. 0.5, Ps suspects that Ad cheats.

In summary, Ps can detect deflation cheating according to C_f and I_f. With a chosen ρ, the expected $k = \hat{C} - C_f$ is at least $(1-\rho)/\rho$. If k is significantly less than its expected value, Ps calculates χ. Figure 4 plots the relation between k and χ for $d = 500$ and different ρ, ξ. It shows that if Ad under-reports $\hat{C}$, a smaller k will result in a larger χ. In the worst case, $\rho = 0.01$ and $\xi = 0.03$ where the expected k is 99, Ad can hide about 80 transactions if the threshold for χ is 0.5. In the best case, $\rho = 0.05$ and $\xi = 0.1$ where the expected k is only 19, any deflation will be detected since χ is almost 1 when $k < 19$.

Note that ξ is application specific. Ps can only tune ρ for the desired security strength. Next, we show how to tune ρ by taking all factors into consideration.

Tuning Security and Cost On the one hand, Ps prefers a larger ρ so that more users return their receipts, and as a result, the chance of successful fraud detection is higher; on the other hand, a larger ρ implies a higher financial cost for more rewards. We show below how Ps strikes a balance between security

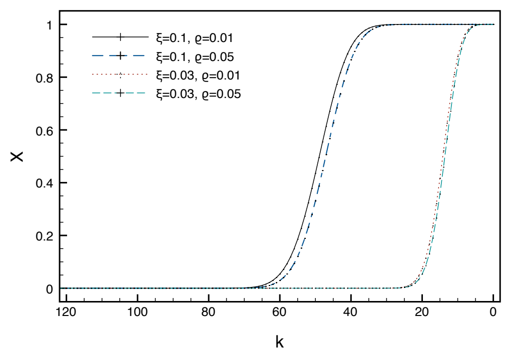
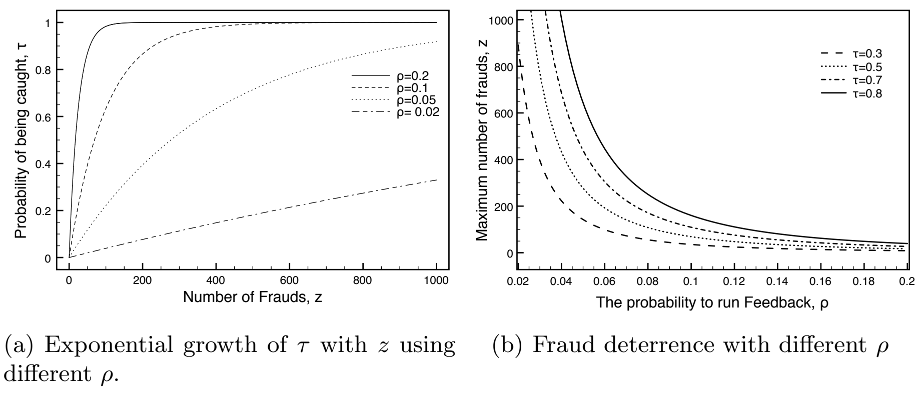


Fig. 4. χ grows when k decreases, for $d = 500$, $\xi = 0.1, 0.03$, and $\rho = 0.01, 0.05$.

and cost by tuning ρ.[3] Let τ denote the probability to successfully catch the dishonest Ad by protocol execution, i.e. $\tau = 1 - (1 - \rho^2)^z$.



(a) Exponential growth of τ with z using different ρ.

(b) Fraud deterrence with different ρ

Fig. 5. Balancing security and cost by tuning the redemption probability

Figure 5(a) depicts the exponential correlation between τ and the number of frauds z, when different fixed ρ are chosen by Ps. As evident in the figure, a low ρ results in a reasonably high probability in detecting a small amount of deflation frauds. Note that in a large scale web advertising, the volume of transactions is usually in thousands or even more. With around one tenth of all users run Feedback, it is expected for our scheme to catch any fraud which accounts for more than one percent of the total volume.

[3] The more rewards are offered by Ps, the higher ρ for user redemption. The exact relation between ρ and financial cost is beyond the scope of this paper.

Suppose that Ad sets up a risk threshold τ and will not make frauds causing a risk higher than τ. Figure 5(b) shows the maximum frauds Ad would have in different risk profiles. The figure depicts the correlation between z and ρ with respect to different constant risk profile τ. For instance, if 10% of the users redeem their receipts, Ad can only cheat around 80 transactions to keep its risk below 50%.

In summary, the anomaly detection has a weaker demand for ρ, than duplication detection does. Therefore, Ps can set ρ according to its own risk profile. Generally, for applications with thousands of transactions, Ps only needs to set ρ between 0.05 and 0.1, which provides sufficient security assurance.

6.3 Other Security Properties

Unforgeability and Non-repudiation. A malicious publisher Ps may collude with corrupted users to forge Ad's signatures so as to frame Ad. We show that the Schnorr signature from Ad is still existentially unforgeable against chosen message attacks. In other words, our scheme does not compromise the security of the standard Schnorr signature scheme.

Theorem 3. *With the discrete logarithm assumption in $\mathbb{G}$ and $\mathbb{G}_h$, Ad's Schnorr signature in Transaction is secure against existential forgery attacks under the random oracle model.*

Proof. (sketch) Let $\mathcal{A}$ be the algorithm forging Ad's Schnorr signature. $\mathcal{A}$ is allowed to access a signature oracle $\mathcal{O}_s$ with a query m, and is allowed to query a random oracle $\mathcal{O}$. $\mathcal{A}$'s goal is to forge a signature (γ^*, β^*) on a message m^* which is not sent to $\mathcal{O}_s$. We show that if $\mathcal{A}$ succeeds in forging a signature, we can construct an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve the discrete logarithm problem in $\mathbb{G}$.

$\mathcal{B}$ is given (g, y, p, q, q') where p, q, q' are large primes satisfying $q = 2q' + 1$ and $q|p-1$, and g's order in $\mathbb{Z}_p^*$ is q. $\mathcal{B}$'s goal is to find x such that $y = g^x \bmod p$. It simulates $\mathcal{O}$ and $\mathcal{O}_s$ and interacts with $\mathcal{A}$. $\mathcal{B}$ sets Ad's public key as y. Then it randomly chooses $h \in \mathbb{Z}_q$, picks $u \in \mathbb{Z}_{q'}$ and computes $v = h^u \bmod q$. $\mathcal{B}$ initializes $\mathcal{A}$ with (y, v, g, h, p, q). The hash function $\mathcal{H}()$ is modeled as a random oracle.

- When $\mathcal{A}$ queries $\mathcal{O}_s$ with t, M, $\mathcal{B}$'s simulation is done as follows:
  1. Select $\gamma, \beta \in_R \mathbb{Z}_q$; Select $\alpha \in_R \{0,1\}^k$;
  2. Set $\mathcal{O}$ such that $\gamma = \mathcal{H}(t\|y\|g\|M\|g^\beta y^\gamma)$ and $\alpha = \mathcal{H}(t\|\mathsf{Ad})$. Namely, $\mathcal{O}$ stores $(t\|y\|g\|M\|g^\beta y^\gamma, \gamma)$ and $(t\|\mathsf{Ad}, \alpha)$ into its local table.
  3. Select $\psi, \lambda_1, \cdots, \lambda_l \in_R \mathbb{Z}_{q'}$;
  4. For $i = 1, \cdots, l$, set $w_i = h^{\lambda_i}$ and $w_i' = g^{\alpha^{\lambda_i}}$ if $\psi[i] = 0$; otherwise set $w_i = v^{\lambda_i}$ and $w_i' = (g^\beta y^\gamma)^{\alpha^{\lambda_i}}$;
  5. Set $\mathcal{O}$ such that $\psi = \mathcal{H}(v\|h\|g^\beta y^\gamma\|g\|t\|w_1\| \cdots \|w_l\|w_1'\| \cdots \|w_l')$, and store $(v\|h\|g^\beta y^\gamma\|g\|t\|w_1\| \cdots \|w_l\|w_1'\| \cdots \|w_l', \psi)$ in the local table and then return $(\alpha, \gamma, \beta, \lambda_1, \cdots, \lambda_l)$ to $\mathcal{A}$.

Note that $\mathcal{A}$ is not able to distinguish whether the tuples returned by $\mathcal{B}$ are simulated results or from a real protocol execution, because all are from the same uniform distribution. Note that $g^{\beta}y^{\gamma}$ in our construction is random as well, because α is an output from the random oracle.

– When $\mathcal{A}$ queries $\mathcal{O}$ with a query m, $\mathcal{B}$ searches its table. If there is an entry (m, m_h), it returns m_h to $\mathcal{A}$. If no such entry is found, $\mathcal{B}$ randomly picks $m_h \in_R \{0,1\}^k$, returns m_h and stores (m, m_h) into the table.

Finally, $\mathcal{A}$ halts and outputs a Schnorr signature $(\gamma^*, \beta^* = r^* - xh^*)$. By the Forking Lemma, $\mathcal{A}$ can produce another valid signature $(\gamma_1^*, \beta_1^* = r^* - xh_1^*)$ with a non-negligible probability by rewinding the random oracle. Therefore, $\mathcal{B}$ can solve the discrete logarithm problem by computing $x = (\beta_1^* - \beta^*)/(h^* - h_1^*)$. $\square$

Theorem 3 shows that neither Ps nor users can forge Ad's Schnorr signatures. This has twofold implications. On the one hand, it provides security assurance for Ad since its signatures are not be forged; on the other hand it implies that Ad cannot repudiate its fraud, if it is caught with two Schnorr signatures using the same hash token.

Privacy. If Ad uses hash tokens along the hash chain, a user knows the count of transactions. In case the exposure of transaction count is undesirable for Ad, the following minor revision on the protocol can be applied.

In Initialization, both Ad and Ps can agree on a pseudo-random permutation function F_s keyed by a shared secret s. For $C \in [1, n]$, $F_s(C)$ maps C into another random number in $[1, n]$. In Transaction, Ad picks a hash token based on $F_s(C)$. Without knowledge of s, $\mathcal{U}$ cannot infer the total amount of hash tokens used by Ad. In Feedback, Ps recovers C using the hash token and $F_s^{-1}(C)$. Note that the revelation of unused hash tokens poses no threat to either Ad or Ps.

6.4 Performance

The amount of information stored by Ad are only its secret keys (x, u), a transaction counter C and a hash chain seed T. Therefore, the storage cost is constant with respect to the amount of transactions. In terms of computation cost, Ad makes $3l + 1$ modular exponentiations. Nonetheless, if Ad needs to reduce its real-time response time, it can pre-compute all the modular exponentiations, then only performs modular additions and hash computations to issue receipts. In terms of communication cost, Ad has a constant communication cost with Ps during Initialization and Report, and sends $(l + 3) \cdot |q|$ bits in Transaction, which is nearly two kilobytes in a practical setting.

The main cost of Ps is its storage overhead for Λ, whose expected size is asymptotically linear with ρN. The main cost for users is their computations for signature verification in Transaction, which involve $3l + 1$ modular exponentiations. Therefore, our scheme may not be suitable for devices with constraints in computation resource, such as mobile phones.

7 Conclusion

In conclusion, we propose a deflation fraud detection scheme for the CPA advertising model, taking a hybrid approach based on cryptography and probability techniques. For any z amount of deflation cheating in the advertiser Ad's transactions, the web publisher Ps can detect it with a success probability at least $1-(1-\rho^2)^z$. As a deterrence, Ps can extract Ad's secret signature key. Although in the long run Ad does not benefit from the deflation fraud by withholding transactions, Ps can still detect it if the amount is over a prescribed threshold. Furthermore, Ps can estimate the expected transaction amount N as $C_f + (1-\rho)/\rho + z_0$ where z_0 is estimated according to Ad's risk profile.

The proposed scheme preserves the simplicity of the existing advertising model, without introducing any third party. It is user-friendly in the sense that users are not required to possess any secrets. Ad's communication, computation, storage cost are all constant. In addition, Ps can tune its security parameter ρ to balance the security and cost.

Acknowledgement

We thank Robert Deng, Jian Wen and Junzhuo Lai for the valuable discussions. We thank anonymous reviewers for their constructive comments. This work is supported by the Office of Research, Singapore Management University.

References

1. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and veriably encrypted signatures from bilinear maps. In *Proceedings of Advances in Cryptology - EUROCRYPTO*, 2003.
2. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *Proceedings of Advances in Cryptology- ASIACRYPTO*, 2007.
3. S. Brands. Untraceable off-line cash in wallet with observers. In *Proceedings of Advances in Cryptology - CRYPTO'93*.
4. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology- CRYPTO*, 1997.
5. D. Chaum. Secret-ballot receipts: True voter-verifiable elections. In *IEEE Security and Privacy*, 2004.
6. D. Chaum, P. Y. Ryan, and S. Schneider. A practical voter-verifiable election scheme. In *Proceeding of ESORICS*, 2005.
7. P. T. Eugster, P. A. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35, June 2003.
8. M. Franklin and D. Malkhi. Auditable metering with lightweight security. *Journal of Computer Security*, 6(4), 1998.
9. M. Gandhi, M. Jakobsson, and J. Ratkiewicz. Badvertisements: Stealthy click-fraud with unwitting accessories. *Anti-Phishing and Online Fraud, Part I Journal of Digital Forensic Practice*, 1, Nov 2006.
10. R. Johnson and J. Staddon. Deflation-secure web metering. *International Journal of Information and Computer Security*, 1, 2007.

11. A. Juels, S. Stamm, and M. Jakobsson. Combatting click fraud via premium clicks. In *Proceedings of USENIX Security*, 2007.
12. M. Kuhn. Probabilistic counting of large digital signature collections. In *USENIX Security Symposium*, 2000.
13. A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *Proceeds of Financial Crypto*, 1998.
14. S. Majumdar, D. Kulkarni, and C. Ravishankar. Addressing broker violations of count integrity in publish-subscribe systems. In *Proceedings of ACM Infocom*, 2007.
15. B. Masucci and D. Stinson. Efficient metering schemes with pricing. *IEEE Transactions on Information Theory*, 47(7), 2001.
16. T. Moran and M. Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *Proceedings of Advances in Cryptology- CRYPTO*, 2006.
17. M. Naor and B. Pinkas. Secure and efficient metering. In *Proceedings of Advances in Cryptology -EUROCRYPTO*, 98.
18. W. Ogata and K. Kurosawa. Provably secure metering scheme. In *Advances in Cryptography – Asiacrypto*, 2000.
19. C. P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology- CRYPTO*, 1989.
20. L. Zhang and Y. Guan. Detecting click fraud in pay-per-click stream of online advertising network. In *Proceedings of ICDCS*, 2008.

Appendix A

Theorem 4. *Suppose that each user has a probability ρ to redeem a receipt. Let P_k denote* Ad*'s escape probability of cheating k times by reusing k distinct hash tokens. Let P'_k denote* Ad*'s maximum escape probability of cheating k times by reusing k' distinct hash tokens, $0 < k' < k$. Therefore, if $\rho < 1/3$, then for all $k > 1$, $P_k > P'_k$.*

Proof. We prove the theorem by using induction on k.
(i) $\mathbf{k = 2}$. It is straightforward to see that $P_2 = (1-\rho^2)^2$. In addition, we have $k' = 1$. In other words, Ad uses a hash token for three times, two out of which are considered as cheating. Therefore, Ad's fraud can be detected when any two of the three corresponding users redeem their receipts. Thus, $P'_2 = (1-\rho)^3 + 3\rho(1-\rho)^2$. So, $P_2 - P'_2 = \rho^2 - 3\rho^3 = \rho^2(1-3\rho) > 0$.
(ii) $\mathbf{k > 2}$. (Induction hypothesis) Suppose that $P_k > P'_k$ for some k. We prove that $P_{k+1} > P'_{k+1}$. Clearly, we have $P_{k+1} = (1-\rho^2)^{k+1}$. Let U denote any user among the $k+1$ cheating victims and t denote the hash token in the signature U receives. Suppose that there are c cheats whereby the token t is used. Therefore, Ad can escape detection when these c cheats are not detected and the remaining $k+1-c$ cheats are not detected either. Hence, If $c = 1$, $P'_{k+1} = P'_k P_1 < P_k P_1 = P_{k+1}$, due to the induction hypothesis. Otherwise, $P'_{k+1} \leq P'_{k+1-c} P'_c < P_{k+1-c} P_c = P_{k+1}$. Thereby, $P'_{k+1} < P_{k+1}$ which completes the proof. □