AccountTrade: Accountable Protocols for Big Data Trading Against Dishonest Consumers

Taeho Jung[1], Xiang-Yang Li[2], Wenchao Huang[2], Jianwei Qian[1], Linlin Chen[1], Junze Han[1], Jiahui Hou[1], Cheng Su[2]
[1]Illinois Institute of Technology, [2] University of Science and Technology of China

***Abstract*—We propose AccountTrade, a set of accountable protocols, for big data trading among dishonest consumers. To secure the big data trading environment, our protocols achieve book-keeping ability and accountability against dishonest consumers who may misbehave throughout the dataset transactions. Specifically, we study the responsibilities of the consumers in the dataset trading and design AccountTrade to achieve accountability against the dishonest consumers who may try to deviate from their responsibilities. Specifically, we propose *uniqueness index*, a new rigorous measurement of the data uniqueness, as well as several accountable trading protocols to enable data brokers to blame the dishonest consumer when misbehavior is detected. We formally define, prove, and evaluate the accountability of our protocols by an automatic verification tool as well as extensive evaluation in real-world datasets. Our evaluation shows that AccountTrade incurs negligible constant storage overhead per file (<10KB), and it is able to handle 8-1000 concurrent data uploading per server depending on the data types.**

I. Introduction

The number of data trading platforms/services has increased with rapidity (*e.g.,* CitizenMe, DataExchange, Datacoup, Factual, Qlik, XOR Data Exchange, Terbine). These data trading platforms, also known as *data brokers*, provide B2C or C2C datasets sales services, and they are the counterparts of physical commodities trading brokers, *e.g.,* Ebay and Amazon. Such a trend has emerged primarily because the sale of digital data became a promising business with the advent of big data [1]. Despite a large business opportunity, the entities possessing large-scale datasets have not readily participated in the data trading due to many security/privacy concerns, and one of them is the lack of accountability. On the one hand, dataset owners are concerned that brokers may illegally disclose or resell their datasets, and on the other hand, they are also concerned ordinary consumers may also illegally trade the purchased datasets. It is possible to deal with the first issue because the accountability of data brokers have already been one of the main concerns of FTC [15], and FTC has managed to detect and punish several data brokers already [2]–[4]; and 2) existing works in the literature can be used to achieve accountability in various systems [20], [25], which make live monitoring on data brokers doable. However, the second issue is non-trivial to address. In general, it is reasonable to enforce such monitoring at the broker's side (as seen from the recent movement [15]), but it is neither justifiable nor practical to monitor the consumers. Firstly, it is unacceptable to lively monitor their behavior because this will raise privacy implications. Secondly, even if the privacy concern is not an issue, the history of Internet strongly suggests that a service mandating the installation of a heavy accountability system will be eliminated due to bad user experiences.

In this paper, we present AccountTrade, a set of accountable protocols for big data trading via data brokers. It enables data brokers to achieve trading-related accountability against dishonest consumers by blaming them when misbehavior is detected. The trading-related misbehavior detected by AccountTrade includes **tax evasion, denial of purchase**, and **resale of others' datasets**. Note that AccountTrade does not try to detect concept-wise or idea-wise plagiarism (*e.g.,* novels with similar plots, images taken with similar concepts, videos taken with similar technique) because whether it is original is a subjective indicator that is hardly decidable. Instead, it detects the *blatant copy* (not an *exact* copy) in uploaded datasets by examining whether the submitted datasets are derived from existing ones. Notably, the copy-detection in table-type datasets has not been studied yet to our best knowledge, and AccountTrade is the first to propose a feasible mechanism. The extra overhead imposed to ordinary consumers is negligible when compared to that of uploading/downloading datasets, and the extra overhead imposed to the data brokers is also acceptable. We formally defined the accountability model in the trading, and we performed automatic proof with formal language as well as theoretic analysis to formally prove the accountability achieved by AccountTrade.

Several challenges make it non-trivial to design AccountTrade. Firstly, the boundary for legal/illegal sale is hard to clearly define. This is partly because dishonest sellers may bring various perturbation into others' datasets before trying to resell them, and defining to what extent data should be perturbed to become *independent* from the original one is not in the computer science domain. Secondly, data brokers manage a large number of datasets with huge volume, but the illegal resale monitoring inherently involves scan (in a certain form) over all datasets that they possess. Finally, as aforementioned, AccountTrade should not impose non-negligible overhead to ordinary consumers (*i.e.,* buyers and sellers). We address these challenges by proposing and utilizing *uniqueness index* in AccountTrade, a new rigorous measurement of the

Xiang-Yang Li is the corresponding author.

The research is partially supported by NSF ECCS-1247944, NSF CMMI 1436786, NSF CNS 1526638, National Natural Science Foundation of China under Grant No. 61520106007, 61625205, 61572453, Key Research Program of Frontier Sciences, CAS. No. QYZDY-SSW-JSC002.

data uniqueness which is resilient to derivative dataset resale and efficiently computable in big data trading with large scale.

The contribution of our paper is summarized as follows.

1. We formally define the symbolic and computational accountability models for big data trading, and design the accountable protocols Upload, Examine, and Download that are provably accountable (Proof in §IV).

2. To efficiently detect illegal resale, we propose and utilize the *uniqueness index* which is consistent with existing state-of-the-art in dataset similarity comparison mechanisms for various data types. Notably, no such mechanism is available for table-type datasets yet, and we propose a novel mechanism to efficiently measure the similarity for table datasets.

3. AccountTrade scales very well with the scale of the data brokers, *i.e.,* the number and volume of the datasets they host. Specifically, the extra overhead introduced at the users' side remains constant regardless of the data brokers' scale, and the extra overhead incurred at the brokers' side linearly grows with the number of the datasets only regardless of their volumes.

II. Definitions and Models

A. Data Trading with Brokers

Three entities exist in our model: *brokers*, *sellers*, *buyers*, and they have different trading-related *responsibilities*.

Broker: Brokers provide general shopping services (product listing, description, payment, deliver, *etc.*). Besides, they are in charge of book-keeping for accounting purpose (*i.e.,* recording trading transactions), and they also need to define the rules to specify what type of selling is considered as re-selling.

Seller: Sellers are mandated to sell only the datasets are collected/generated by themselves, and they should not re-sell others' datasets by slightly perturbing them. Also, they should honestly file the tax report regarding the dataset sale, and they are forbidden to disturb the brokers' book-keeping.

Buyer: Buyers should not disturb the brokers' book-keeping.

Some areas are important but orthogonal to ours. Properly describing quality/utility of datasets for buyers [40] is complementary, and sophisticated pricing mechanism may exists but we simply let sellers set the prices.

B. Adversary Model & Channel Assumption

Malicious users: We assume users may try to avoid the aforementioned responsibilities, *e.g.,* disrupting the brokers' data trading service enabled by AccountTrade, denying cleared transactions (*i.e.,* paid and sold), and illegally selling previously purchased dataset. A user is defined as a *dishonest* user if he avoided any of the trading-related responsibility, and such behavior (either selling or buying) is denoted as *misbehavior*. Note that, when illegally selling previously purchased datasets, attackers may try to perturb the dataset to bypass existing copy-detection approaches.

Trusted brokers: We assume the brokers can be trusted, *e.g.,* the role is played by the organizations that are strictly supervised with great transparency or commercial companies with high reputation. Similar assumptions can be found in [32], and the assumption that the brokers will strictly supervised is also consistent with the FTC's recent action [15].

Channel assumption: We assume both buyers and sellers interact with the broker via secure communication channels. The communication is encrypted and decrypted with pre-distributed keys to guarantee that the dataset is not open to the public. This also implies authentication is in place since the broker needs to use the correct entity's key for communication.

C. Accountability Model

We inherit [37] to define our own accountability model.Our accountable protocols are characterized by the two properties:

- *Fairness*: honest entities are never blamed.
- *Goal-centered completeness*: if accountability is not preserved due to malicious entities' misbehavior, at least one of them is blamed.

General *completeness*, which states all misbehaving entities should be blamed, is impossible to satisfy because "some misbehavior cannot be observed by any honest party" [37]. AccountTrade also requires *individual accountability*, which states it must be able to correctly blame one or more parties unambiguously, rather than to blame a group among which some entities misbehaved but do not know whom.

In the sequel, two formal definitions of accountability with different purposes are presented. *symbolic individual accountability* is defined in an ideal setting where all building blocks are abstracted as ideal black-box ones. The symbolic model is amenable to automatic security verification protocols [8] who automatically verify whether there exist security flaws. Then, *computational individual accountability* without the abstraction will be defined to give a more fine-grained measurement of individual accountability in real protocol runs.

Symbolic Individual Accountability: A *verdict* is a boolean formula ψ consisting of the propositions having the form $\mathsf{dis}(e)$, where $\mathsf{dis}(e)$ is a statement that the entity e misbehaved. If the broker states $\psi = \mathsf{dis}(A) \wedge \mathsf{dis}(B)$, it means the broker blames A and B, and the blame is fair if A and B indeed misbehaved. A *run* r is an actual run of a protocol; then we use the expression $r \Rightarrow \psi$ to denote that ψ evaluates to true in the run r. Then, for a run r with misbehavior occurred and the ψ describing the misbehaved entities in r, we call $\phi = (r \Rightarrow \psi)$ an *accountability constraint* of r because the broker must state ψ after observing the run r. We use Φ to denote the set of all accountability constraints of all possible runs in a given protocol P, denoted as P's accountability property. We say an entity J ensures Φ after observing a run r if either no misbehavior occurred in r or J states ψ and $(r \Rightarrow \psi) \in \Phi$.

Definition 1 (Symbolic)**.** *Let P be a protocol, J be its entity, and Φ be P's accountability property. We say P is individually Φ-accountable w.r.t. J if*

- *Fairness: verdicts stated by J all evaluate to true,*
- *Goal-centered completeness: for every run r of P, J ensures Φ after observing it, and*
- *Individual accountability: the only logical operators in J's verdicts are '$\wedge$'.*

Computational Individual Accountability: The computational version is similar to the symbolic one except that we consider that leveraged building blocks may be imperfect. For example, there are always negligible chances for the attacker to break (almost all) cryptographic tools (*e.g.,* by a random guess or with negligible advantage), and the leveraged predictive models can hardly be perfect regarding the precision and recall. By reusing the notations in the symbolic model, we present the following definition.

Definition 2 (Computational)**.** *Let P be a protocol, J be its entity, and Φ be P's accountability property. We say P is individually (Φ, η, χ)-accountable w.r.t. J if*

- *Fairness: for any verdict ψ stated by J, $\Pr[\psi = \mathrm{F}]$ is bounded by η,*
- *Goal-centered completeness: for any run r of P, $\Pr[\neg(J \text{ ensures } \Phi)]$ is bounded by χ, and*
- *Individual accountability: the only logical operators in J's verdicts are '$\wedge$'.*

D. Defining AccountTrade

Upload, $\mathsf{Examine}$, and $\mathsf{Download}$ comprises AccountTrade. The expression $[\{\mathsf{entity} : \mathsf{input}\}] \rightarrow_P [\{\mathsf{entity} : \mathsf{output}\}]$ is used to define the input and output from different entities in the protocol P, and $\perp$ indicates a null argument.

Upload: This protocol is executed between a seller who wishes to sell her dataset d and the broker. The seller generates a post post_t at time t which is posted at the public bulletin board so that the broker can book-keep the transaction and achieve individual accountability.

$$[\text{Seller} : d; \text{Broker} : \perp] \rightarrow_{\mathsf{Upload}} [\text{Seller} : \perp; \text{Broker} : d, \mathsf{post}_t]$$

$\mathsf{Examine}$: The broker examines whether the dataset is derived from existing ones with this protocol. He generates a set of *MinHash* values $mh_\pi(d)$ (defined in §III-B) for the dataset d, and they are used to calculate the uniqueness index of d, $U_{\mathbb{D}}(d)$, over the entire database $\mathbb{D}$ containing all already-uploaded datasets.

$$[\text{Broker} : d] \rightarrow_{\mathsf{Examine}} [\text{Broker} : \{mh_\pi(d)\}_\pi, U_{\mathbb{D}}(d)]$$

$\mathsf{Download}$: This protocol is executed between a buyer and the broker. The buyer generates and posts post_t at the bulletin board similar to Upload protocol.

$$[\text{Buyer} : \perp; \text{Broker} : d] \rightarrow_{\mathsf{Upload}} [\text{Buyer} : d; \text{Broker} : \mathsf{post}_t]$$

E. Design Goal

Let Φ be $\{r \Rightarrow \mathsf{dis}(e) | r \in \Gamma\}$ where Γ is a set of runs containing misbehavior. This paper's goal is to design the protocols such that Upload, $\mathsf{Examine}$, and $\mathsf{Download}$ have both symbolic individual Φ-accountability and computational individual Φ, η, χ-accountability *w.r.t.* the broker.

III. Specifications of AccountTrade

We leveraged several cryptographic building blocks.

Cryptographic hash: Suppose Σ is a set of characters. We leverage a cryptographic hash function $\mathsf{H} : \{0,1\}^* \rightarrow \Sigma^k$ where k is a pre-defined system-wide parameter. The hash function maps any bitstring to a string of length k.

Digital signature: A secure digital signature scheme is leveraged to let an entity E sign on a message $m \in \Sigma^*$. Produced signature is denoted by $\mathsf{sig}_E(m)$, and it is used to verify the integrity of m. We also let every signature secret key be bound to a specific user so that the signature can be used to prove E's ownership for accountability purpose. For the simplicity, we omit the signature verification in the protocol specifications.

Append-only bulletin board: A bulletin board with 'append' and 'read' privileges only [22] has been employed as the source of trust in systems requiring accountability or verifiability [6], [8]. It is a public broadcast channel with memory where any party can post messages by appending them to her own area, and she can see anyone's posts as well. A posted message is denoted as a 'post' hereafter.

A. Upload for Sale

When a seller A wants to upload dataset d to broker to sell it, she follows the Upload protocol (Fig. 1) and posts her declaration post_t at the bulletin board at time t, where '$||$' denotes string concatenation. Then, she initiates the upload request with $\mathsf{H}(d)$, where the hash is applied on d's bitstream. The broker finds the corresponding post from the bulletin board and blames A if none is found, because it is evident that she has tried to avoid being book-kept. If the broker sees the post, he accepts A's request and retrieves the dataset. Then, the broker checks whether the hash of received dataset is identical to the one posted at the bulletin board and blames A if not. Finally, the broker generates the description of the dataset d (*e.g.,* by [40]) and make it public at his trading platform.

B. Dataset Examination

If the upload is successful, the broker further checks whether similar dataset has been uploaded before. To do so, we propose *uniqueness index*, which is indicative of how much of overlaps a given set S has over a set of sets $\mathbb{S} = \{S_1, \cdots, S_n\}$.

Definition 3 (Uniqueness index)**.** *Given a set of $\mathbb{S} = \{S_1, S_2, \cdots, S_n\}$, the uniqueness index of S_x over the set $\mathbb{S}$ is defined as $U_{\mathbb{S}}(S_x) = 1 - \max_{S \in \mathbb{S}}\{\Delta(S, S_x)\}$, where $\Delta(S, S_x)$ is a normalized similarity function describing how unique S_x is when compared to S, defined as:*

$$\Delta(S, S_x) = J(S, S_x) \cdot \frac{\max(|S|, |S_x|)}{\min(|S|, |S_x|)}$$

$J(S_1, S_2)$ refers to the Jaccard Index, which is statistical measurement of the similarity and diversity of two given sets, defined as $J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$. Then, we define selling of a dataset d as re-selling if $U_{\mathbb{D}}(d) > \theta_{\mathsf{high}}$ and as valid selling if $U_{\mathbb{D}}(d) < \theta_{\mathsf{low}}$, where $\mathbb{D}$ is the database of datasets the broker possesses and $\theta_{\mathsf{high}}, \theta_{\mathsf{low}}$ refer to two threshold values for decision making. If the uniqueness index is between the two threshold values, the broker can manually inspect the dataset with human labor.

The reason we are going to use this uniqueness index in dataset re-selling detection is manifold. Firstly, it intuitively measures how many of the elements in S_x are similar to the elements in the entire set $\mathbb{S}$, and the the multiplier after the

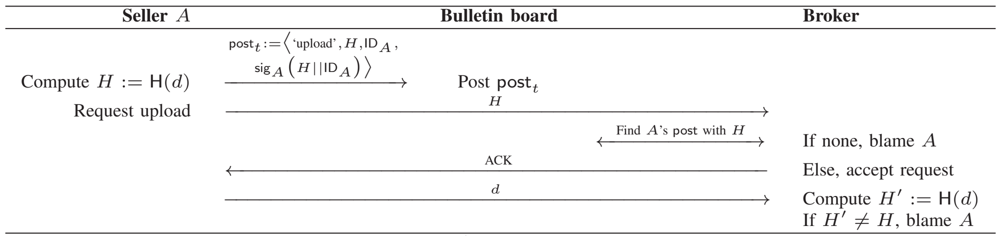

Seller A
Bulletin board
Broker
$\mathsf{post}_t := \langle \text{‘upload’}, H, \mathsf{ID}_A, \mathsf{sig}_A(H||\mathsf{ID}_A) \rangle$
Compute $H := \mathsf{H}(d)$
Post post_t
Request upload
H
Find A's post with H
If none, blame A
ACK
Else, accept request
d
Compute $H' := \mathsf{H}(d)$
If $H' \neq H$, blame A

Figure 1. Upload protocol between a seller A with ID ID_A and the broker for uploading dataset d.

Jaccard Index guarantees the index is equal to 1 when S_x is a subset/superset of any set in $\mathbb{S}$. Secondly, in many existing similarity comparison approaches in information retrieval, the datasets are considered as sets of elements (k-grams for texts, feature descriptors for images, and key frames for videos), and therefore the proposed uniqueness index is consistent with them (reviewed in §VI). As for the tables, there is no known similarity comparison mechanism available yet, but we designed a novel similarity mechanism based on the uniqueness index. Thirdly, our extensive experiments on the real-world datasets (§V-B) shows that the uniqueness index of d_x is extremely low (≤ 0.05) when d_x is derived by applying simple perturbation to existing data, and that the index is as high as $0.8 \sim 1.0$ when d_x is unique. This prunes many illegal re-selling cases automatically with high confidence when the threshold values are selected conservatively. We will elaborate how to choose the thresholds in §V-B. Finally, because of the clearly separable uniqueness indices, it becomes possible to make threshold-based decisions at the broker's side, and it enables the broker to scale well with the number of users as well as the number of datasets owing to the simplicity of the threshold-based decision making.

MinHash: MinHash [10] is a special hash for quick estimation of Jaccard Index of two sets. It is assumed that the universe set of elements that may appear in sets is finite and ordered, and a finite bit vector $\vec{\mathbf{S}}$ is used to represent a set S. That is, $\vec{\mathbf{S}}[i] = 1$ if S has the i-th element in the universe and 0 otherwise. Firstly, a permutation π over $\{1, 2, \cdots, |\vec{\mathbf{S}}|\}$ is randomly selected, and the MinHash value of $\vec{\mathbf{S}}$ for this π (denoted as $mh_\pi(\vec{\mathbf{S}})$ hereafter) is the position of the first bit whose value is 1 when all bits in $\vec{\mathbf{S}}$ are visited with the order given by the permutation π. For example, suppose the size of the universe set is 10 and we are given the membership vector of a set $\vec{\mathbf{S}} = \{0, 1, 0, 0, 0, 0, 0, 1, 1, 1\}$. Then, $mh_\pi(\vec{\mathbf{S}})$ corresponding to the permutation $\pi = (5, 7, 3, 2, 9, 10, 1, 4, 8, 6)$ is equal to 8 because if one probes the 7th bit ($\pi[7] = 1$), 4th bit ($\pi[4] = 2$), 3rd bit ($\pi[3] = 3$), 8th bit ($\pi[8] = 4$) by following the permutation π, the 8th bit is the first 1 he has. The following property from [11] explains why we propose to apply MinHashing: $\Pr[mh_\pi(\vec{\mathbf{S}}_1) = mh_\pi(\vec{\mathbf{S}}_2)] = \frac{||\vec{\mathbf{S}}_1 \wedge \vec{\mathbf{S}}_2||}{||\vec{\mathbf{S}}_1 \vee \vec{\mathbf{S}}_2||}$ for any S_1, S_2. Then, M MinHash values achieved from $\vec{\mathbf{S}_1}$ and $\vec{\mathbf{S}_2}$, each set of M values extracted from the same set of M unique permutations $\{\pi_1, \pi_2, \cdots, \pi_M\}$, can be used to estimate $J(\vec{\mathbf{d}}_1, \vec{\mathbf{d}}_2)$ with $\frac{\text{\# of same } mh}{M}$ because the Chernoff-Hoeffding bound tells the expected error is $O(\frac{1}{\sqrt{M}})$.

If datasets can be converted to sets by finding out how to extract elements from the datasets, and if enough permutations are defined to achieve enough MinHash values (*i.e.,* large M in the error $O(\frac{1}{\sqrt{M}})$), one can easily approximate the uniqueness index of a dataset d given a database of datasets $\mathbb{D}$ with a linear search on it. Any similarity comparison mechanism which treats a dataset as a set of elements is accepted in Account-Trade. To present the feasibility, we present one example for each type of dataset. Notably, because such a mechanism for table and graph types is not available in the literature yet, we present our novel mechanism. For other types, we present fast variants of current state-of-the-art works in the literature.

Text: The text document is *shingled* to k-shingles (also known as k-grams) [11]. A k-shingle is a contiguous sequence of k characters including spaces (may or may not include others) in a text. For example, a text string "bad boy" turns into {"bad ", "ad b", "d bo", " boy"} when shingled to 4-shingles. Then, a text dataset is treated a set of k-grams, and its bit vector can be created when k is fixed (a vector of 27^k dimensions). A good rule of thumb is that $k = 5 \sim 9$ for large documents similarity comparison [33]. To avoid being biased by stop words, they are removed first, and to achieve robustness against trivial perturbation, synonyms are replaced with a pre-defined representative in the family of synonyms (using the synsets in WordNet [31]). Finally, the MinHash values can be extracted from the vectors.

Image: Feature descriptors (*e.g.,* SIFT [30], [41]) are useful in object recognition [24]. They are automatically extracted from the images, which are high-dimensional vectors describing points, edges, or regions in the images. Each image may have hundreds of feature descriptors, and they may be binary, *i.e.,* $\{0, 1\}^d$, or real, *i.e.,* $[0, 1]^d$. Since d is as large as 128, it is impossible to directly turn the list of features to a membership vector in either case. Therefore, we form a finite *feature universe* by extracting features from a set of images whose features are diversely distributed in the feature space (*e.g.,* Flickr1M set [5]), and the bit vector of an image is generated by finding the nearest neighbors in the feature universe.

Video: Keyframes extraction [23] summarizes a video with several *keyframes* which are essentially images. Then, the uniqueness of the keyframes can be evaluated as aforementioned, and the uniqueness indices of keyframes can be aggregated to evaluate the uniqueness index of the video (*e.g.*, the minimum index defined as the uniqueness index of the video).

Table: Retrieval or similarity comparison on large-scale table datasets have not been researched yet. We propose a novel mechanism by leveraging the MinHash and the *bloom filter* [9]. A bloom filter is a vector of m bits representing a set of items. The filter has k uniformly distributed hash functions which map an item to a position in the vector. When an item is inserted to the set, k hash values (*i.e.,* positions) are calculated, and all corresponding bits are set to 1. Then, to query whether an item i_x exists in a set $S = \{i_1, i_2, \cdots\}$, all i_x's hashed positions are probed to see if all bits are 1. If any of the bits is 0, $i_x \notin S$ with zero false negative ratio; if all of these bits are 1, $i_x \in S$ with a bounded false positive ratio. It is known that the false positive ratio is the lowest when $k = \frac{m}{n} \ln 2$ when m, n are given, and the filter size with this optimal k is $m = -\frac{n \ln p}{(\ln 2)^2} \approx 9.58n$ for a given false positive ratio $p = 0.01$ and the number of items n to be inserted [18]. For the sake of simplicity, we use $B^k_m(d)$ to denote d's bloom filter with size m and k hash functions hereafter.

We treat each row as an element to be inserted, and we construct a bloom filter for each table dataset d as $B^k_m(d)$. Then, due to the bloom filters' properties, $B^k_m(d_1 \cup d_2) = B^k_m(d_1) \vee B^k_m(d_2)$ and $B^k_m(d_1 \cap d_2) = B^k_m(d_1) \wedge B^k_m(d_2)$, where the intersection and union are performed on the rows. Owing to the fact that the approximate number of items that have been inserted in $B^k_m(d)$ is $f(|B^k_m(d)|) = -\frac{m \ln(1-|B^k_m(d)|/m)}{k}$ where $|B^k_m(d)|$ denotes the number of 1's in $B^k_m(d)$ [36], the Jaccard Index of d_1, d_2 is approximated by

$$J(d_1, d_2) = \frac{|d_1 \cap d_2|}{|d_1 \cup d_2|} \approx \frac{f(|B^k_m(d_1) \wedge B^k_m(d_2)|, m, k)}{f(|B^k_m(d_1) \vee B^k_m(d_2)|, m, k)}$$

One drawback is that the above method only addresses horizontal partitioning. Therefore, we insert all subsets of every row so that the vertical partitioning is also covered by the bloom filter. That is, for a table having c columns, we insert $2^c - 1$ different sub-tuples for each row to the table's bloom filter, which leads to $(2^c - 1)r$ items inserted into the bloom filter when there are r rows in the table. Because of the performance concern in the relational database management (*i.e.,* join operations, dynamic rows), the number of columns is usually small (*e.g.,* less than 10). Therefore, AccountTrade can apply this approach for the majority of the tables. If the table's column c is so large that $(2^c - 1)r$ is too large, we compress the number of elements in the bloom filter from $O(2^c r)$ to $O(c^g r)$ at the cost of accuracy loss, where g is a constant. Namely, instead of inserting all sub-tuples, we insert the tuples of size $1, 2, 3, \cdots, g$ only. This translates to the insertion of $\binom{c}{1}, \binom{c}{2}, \binom{c}{3}, \cdots, \binom{c}{g}$ tuples only, which is $O(c^g)$. With this approximation, if every attribute in the table (both row-wise and column-wise) is unique, the false positive ratio of a row-query when $c > g$ is $1 - (1-p)^{\lceil c/g \rceil}$ where p is the false positive ratio of the bloom filter.

The second drawback is that, even though the bloom filter is compressed from $O(2^c r)$ to $O(c^g r)$, the filter size m can be very large since a large m will lead to a lower false positive ratio (9.58 bits per item is required to achieve 1% false positive rate). Then, overhead of the bit-wise OR and AND operations incurred in the approximation of $J(d_1, d_2)$ may be significant. We cannot directly use MinHash to approximate the uniqueness index because the Jaccard Index of two bloom filters is different from the Jaccard Index of the original tables. However, we can indirectly approximate the uniqueness index in a **constant time** irrelevant to m with the following trick. Recall that we can estimate $J_{12} = \frac{|B^k_m(d_1) \wedge B^k_m(d_2)|}{|B^k_m(d_1) \vee B^k_m(d_2)|}$ with MinHash values of $B^k_m(d_1)$ and $B^k_m(d_2)$. Therefore if we can calculate $|B^k_m(d_1) \wedge B^k_m(d_2)|$ or $|B^k_m(d_1) \vee B^k_m(d_2)|$ from the estimated J_{12}, we are able to estimate $J(d_1, d_2)$ based on the bloom filters regardless of m. Normally it is not possible to calculate X, Y from $\frac{X}{Y}$ only, but we can derive them as follows by using the inclusion-exclusion principle: $|B^k_m(d_1) \vee B^k_m(d_2)| = \frac{|B^k_m(d_1)| + |B^k_m(d_2)|}{1 + J_{12}}$, $|B^k_m(d_1) \wedge B^k_m(d_2)| = J_{12} \cdot \frac{|B^k_m(d_1)| + |B^k_m(d_2)|}{1 + J_{12}}$. To calculate them in a constant time, we augment the bloom filter and use an extra field to store $|B^k_m(d)|$ for each filter so that this field can be accessed in a constant time. Then, if we use all same augmented bloom filters $B^k_{m,\mathsf{aug}}(\cdot)$ for all table datasets, AccountTrade can approximately calculate the Jaccard Index $J(d_1, d_2)$ in a constant time.

Graph: If graphs are unlabeled, *i.e.,* nodes and edges do not have any attribute, the similarity comparison between two graphs involves the sub-graph isomorphism problem, which is NP-complete [16]. Many mechanisms have been proposed to solve this problem with graph indexing, but most of them cannot be applied in our scenario managing big graphs of millions of nodes. The most efficient mechanism in the literature from Sun *et al.* [35] generates 6 GBytes index and 33s index time for a graph of 1 billion nodes, but even in their work, the query time is several seconds for a query graph with only tens of nodes and edges. This is due to the inherent hardness of the sub-graph isomorphism problem. Therefore, we only consider the labeled graphs where nodes or edges have attributes, and focus on evaluating the attribute-wise similarities/uniqueness. Graphs are stored as tables, therefore we treat the tables as tables and use the same method as aforementioned.

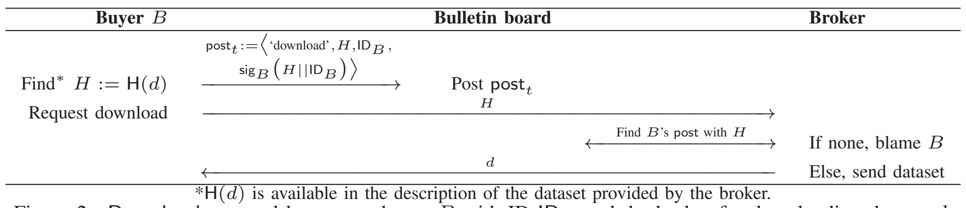
After the uploaded dataset's uniqueness index is calculated with a linear scan on $\mathbb{D}$, the broker can automatically decide whether he will accept it (if the index is very high), reject it (if the index is very low), or leave it to manual inspection (otherwise). How to set up the thresholds will be explained in §V-B. Although the complexity of the uniqueness index calculation is $O(|\mathbb{D}|)$, the cost of the linear scan is negligible (2.5ms per million datasets in $\mathbb{D}$ in our setting, §V-A2).

C. Download after Purchase

When a buyer B wishes to get access to certain dataset d (after reading the description provided by the broker), she pays for it to the broker first and then follows the Download protocol (Fig. 2). She first posts a declaration post_t at the bulletin board at time t, and she initiates the download request by sending $\mathsf{H}(d)$ to the broker, where $\mathsf{H}(d)$ is available in the description of the dataset provided by the broker. The broker finds the corresponding post from the bulletin board and blames B if none is found, because it is evident that she has tried to avoid being book-kept. If the broker sees the post, he accepts B's download request and sends the dataset to B.

D. Parallelization

The most intensive overhead comes from the 1) data file I/O; 2) conversion to membership vector; and 3) generating M



*$\mathsf{H}(d)$ is available in the description of the dataset provided by the broker.

Figure 2. Download protocol between a buyer B with ID ID_B and the broker for downloading dataset d.

MinHash values. Therefore, we consider reducing the overall execution time by introducing parallelization. Data is logically partitioned into chunks so that each processor only reads the designated chunk. We carefully design the partitioning among processors so that the membership vector will not be incomplete due to the partitioning. Finally, we compute M MinHash values in parallel since each value is independent of another. Note that processors can read one file simultaneously in 'Read Only' mode, and that both Windows and Linux file systems support random access.

Text: The document is divided into several chunks such that two consecutive chunks have k overlapped letters. Then, each processor individually and independently loads the k-shingles into the shared membership vector simultaneously.

Image: The image is divided into chunks of same resolution (each chunk is a cropped image) where adjacent chunks share as many pixels as the range of the feature descriptor, and each processor extracts features in each chunk. Then, each processor finds the nearest neighbor in the visual word space and fill it into the shared membership vector simultaneously.

Video: In most works, the keyframes are extracted based on the temporal differences in a time window of t frames, therefore the video is divided into chunks where t frames are overlapped between consecutive chunks. Then, the same procedure as in image-type data can be followed.

Table & Graph: The table is horizontally partitioned into several chunks of the same number of columns. Each processor can independently load each item (*i.e.,* sub-tuples) to the shared bloom filter simultaneously.

E. Accountability Properties of AccountTrade

Upload

J1: If the post post_t matching H does not exist, the broker states $\mathsf{dis}(A)$ where A is the one who sent the upload request.

J2: If the posted hash H in post_t is different from the calculated hash H', the broker states $\mathsf{dis}(A)$.

Examine

J3: If the calculated uniqueness index is very low or the manual inspection indicates the dataset is derived from already-uploaded ones, the broker states $\mathsf{dis}(A)$ where A is the one who uploaded the dataset.

Download

J4: Same as **J1** except that $\mathsf{dis}(B)$ is stated instead, where B is the one who sent the request.

J1 detects a dishonest seller who tries to deny a sale transaction, and J2 further prevents a dishonest seller from declaring a wrong dataset. J3 detects reselling, and J4 detects a dishonest buyer who tries to deny a purchase transaction. Due to the aforementioned judge processes, sellers are not able to avoid being taxed for their dataset sale, and buyers cannot resell the purchased datasets or deny the purchase. Therefore, AccountTrade successfully achieves the pre-defined accountability.

IV. Proof of Accountability

A. Automatic Proof for Symbolic Model

We formally verify the fairness and completeness in Upload and Download protocol by using ProVerif [8], an automatic symbolic protocol verifier. Proverif models a protocol by a group of parallel processes. Each step in a process is a statement of the form $\mathbf{in}(c, m)$ or $\mathbf{out}(c, m)$, meaning that a message m is received from or sent to the channel c. After modelling the processes, ProVerif automatically verifies the soundness of the protocol by validating predicates that should be satisfied (various accountability properties in our case). We additionally model and enumerate all possible misbehavior from seller A and buyer B, and we also modelled two types of processes for each of two entities: honest/dishonest sellers and honest/dishonest buyers. Honest processes strictly follow the protocols and dishonest processes enumerates all misbehavior. Then, we add events for the broker to blame dishonest participants when it detects one. For the seller A, the honest/dishonest process runs in parallel, and we add an event $\mathsf{event}(\mathbf{NFol}(A))$ before the dishonest process executes. Then, in the case the broker finds A has performed misbehavior, the event $\mathsf{event}(\mathbf{JNFol}(A))$ is executed. Hence, to ensure that the broker has made correct decisions, we validate the predicate by using the automatic verification function in ProVerif: $\forall x.\mathsf{event}(\mathbf{JNFol}(x)) \Rightarrow \mathsf{event}(\mathbf{NFol}(x))$. Specifically, the soundness in case **J1**, **J2**, **J4** are verified by using the predicate. The model of AccountTrade is available at "goo.gl/OWai5W", and it passed the automatic verification by ProVerif. This indicates AccountTrade's design is flawless.

B. Theoretic Proof for Computational Model

Recall that a uniqueness index is approximated with M MinHash values. Let κ be the security parameter, $\varepsilon_{\mathsf{conv}}$ be the upper bound of the error introduced in dataset-to-set conversion, and $\Pr[\varepsilon_{\mathsf{conv}}] = 1 - \delta_{\mathsf{conv}}$ be the probability that this worst case occurs. Then, we have Theorem 1.

Theorem 1. *Given an accountability property Φ, the broker in AccountTrade computationally and individually ensures* $\big(\Phi, \delta_{mh}+\delta_{\mathsf{conv}}-\delta_{mh}\delta_{\mathsf{conv}}, \max(\frac{1}{2^\kappa}, \delta_{mh}+\delta_{\mathsf{conv}}-\delta_{mh}\delta_{\mathsf{conv}})\big)$*-accountability, where* $\delta_{mh} = 2\exp(-\frac{M(\theta_{\mathsf{high}}-\theta_{\mathsf{low}}-\varepsilon_{\mathsf{conv}})^2}{2})$.

Proof. η in **Fairness**: In **J1**, it is impossible that an honest entity will be blamed as the hash values are always calculated correctly. Therefore, $\eta_{\mathrm{J1}} = \Pr[\mathsf{dis}(A) = F] = 0$ in **J1**.

Similarly, $\eta_{J2} = \eta_{J4} = 0$. η_{J3} in **J3** is related to the re-selling detection in Examine protocol. AccountTrade defines a dataset d as valid if $U_{\mathbb{D}}(d) > \theta_{\mathsf{high}}$ and illegal otherwise. Then, an honest seller is blamed when the calculated uniqueness index is below θ_{low} but the true index should have been above θ_{high}. Let $\hat{U}_{\mathbb{D}}(d)$ be the uniqueness index approximated by M MinHash values and $U_{\mathbb{D}}(d)$ be the true uniqueness index. The Chernoff-Hoeffding bound tells $\Pr[|\hat{U}_{\mathbb{D}}(d) - U_{\mathbb{D}}(d)| < \varepsilon_{mh}] > 1 - \delta_{mh}$, when $M = \frac{2}{\varepsilon_{mh}^2} \ln \frac{2}{\delta_{mh}}$ for any constant $\varepsilon_{mh}, \delta_{mh}$. Then, an honest seller is blamed when sum of two errors exceed $\theta_{\mathsf{high}} - \theta_{\mathsf{low}}$, *i.e.,* $\varepsilon_{mh} > \theta_{\mathsf{high}} - \theta_{\mathsf{low}} - \varepsilon_{\mathsf{conv}}$, whose probability is bounded by $\eta_{J3} = 1 - (1 - \delta_{mh})(1 - \delta_{\mathsf{conv}})$. In order not to have false blaming, ε_{mh} should be no greater than that, in which case $\delta_{mh} = 2\exp(-\frac{M(\theta_{\mathsf{high}} - \theta_{\mathsf{low}} - \varepsilon_{\mathsf{conv}})^2}{2})$. In conclusion, $\eta = \max(\eta_{J1}, \cdots, \eta_{J4}) = \eta_{J3}$.

χ in **Completeness**: In **J1**, if hash collision occurs for $d' \neq d$ (*i.e.,* $\mathsf{H}(d) = \mathsf{H}(d')$), a dishonest seller becomes able to request uploading a new dataset d without being book-kept. However, the probability of this is as small as $\frac{1}{2^\kappa}$ where κ is the security parameter, therefore $\chi_{J1} = \frac{1}{2^\kappa}$ where $\chi_{J1} = \Pr[\neg(\text{Broker ensures } \Phi)]$ for a run with dishonest seller in **J1**. Similarly, $\chi_{J4} = \chi_{J2} = \frac{1}{2^\kappa}$. Analysis similar to that in the fairness above also holds for χ_{J3}, and $\chi_{J3} = \eta_{J3} = \delta_{mh} + \delta_{\mathsf{conv}} - \delta_{mh}\delta_{\mathsf{conv}}$. Therefore, $\chi = \max(\frac{1}{2^\kappa}, \delta_{mh} + \delta_{\mathsf{conv}} - \delta_{mh}\delta_{\mathsf{conv}})$. □

V. Validation on Prototype Testbed

We acquired texts, images, tables, and graphs datasets from publicly available sources. Text datasets are from [27], [28]; image datasets are from [5], [38]; video datasets are from [7], [19]; and table & graph datasets are from [27], [28]. Also, AccountTrade's programs are deployed in a server with Intel(R) Xeon E5-2620 and 32GB DDR4 1866.

A. Microbenchmark with Real Data

Note that we implemented parallelized algorithms (§III-D).

1) Extra overhead of Upload *and* Download*:* The extra information AccountTrade requires brokers to store only includes the MinHash values. Per each published file, brokers only need to store M MinHash values (1024 long integers in our simulation, translating to 8 KB). Sizes of IDs, hash values, and the signature in the post are fixed and negligible. Therefore, the overall extra communication overhead among the consumers, the bulletin board, and the broker is negligible.

2) Run time of Examine*:* Examine consists mainly of uniqueness index calculation, whose different benchmarking results are shown in Fig. 3. Fig. 3(a)-(e) present the time consumption in calculating 1024 MinHash values from different types of data. All the run time presented in Fig. 3 include the I/O overhead. The last step of uniqueness index calculation is finding the maximum of $\Delta(S, S_x)$ (Def. 3) over the entire database. This step is not type-dependent, and it is no more than linear search across the database which incurs 2.5ms per million datasets at the broker's side.

It is noticeable that MinHashing time does not change much when the size of the data increases in all types except Video (whose MinHashing is not individually shown in the figure). This is natural since MinHashing requires AccountTrade to permute on the membership vector until it finds the first bit that is 1, and in theory the run time of it should be inversely proportional to the number of items in the membership vector, which implies the MinHashing time is inversely proportional to the data size as well. The uniqueness index calculation in the video type data (Fig. 3(c)) involves MinHashing, but the number of times MinHashing is performed is proportional to the number of frames, and this is why its green bars grow with the frames. In our implementation, we set the threshold for sub-tuple approximation in the table data as 7 columns, and this is why run time grows exponentially until 7 columns and then polynomially after it in Fig. 3(e).

Examine is executed when sellers upload their dataset, and it does not have to be performed in a real time since it is usual that a hold is placed on an uploaded dataset in real dataset trading platforms. Therefore, the presented benchmark results are promising in that most of the data can be examined in a time that is negligible to the uploading time.

B. Large-scale Simulation with Real Data

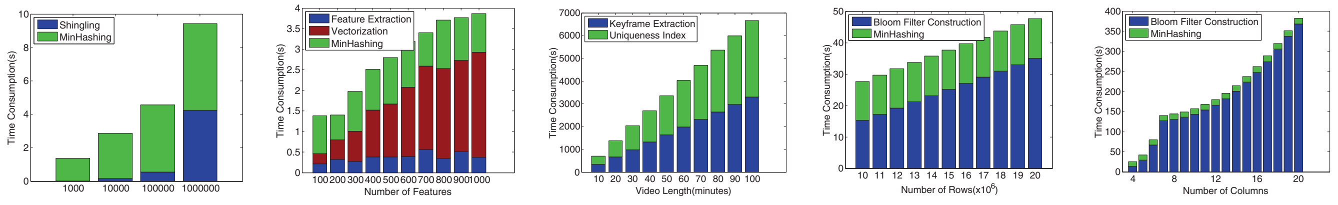
We analyzed the distribution of uniqueness indices in real-world datasets to explore whether the uniqueness index space $[0, 1]$ can be clearly divided into two areas: those for legal sale and those for illegal re-sale. We created a database of data files containing MinHash values of 5,000 data files for each type except video type which contains MinHash values of 500 files. Then, we acquired a ground truth datasets of *unique data* and *derivative data* as follows. We considered data A from a different source from B as unique from B and vice versa, and we considered data A as derivative if it is achieved by applying the following perturbation on B.

- Text: Replace words with synonyms; switch active/passive tense; delete sentences; merge documents.
- Image & video: Crop the image/video; rotate the image/video; merge existing images/video.
- Table & graph: Delete rows; delete columns.

The consequent distributions are shown in Fig. 4.

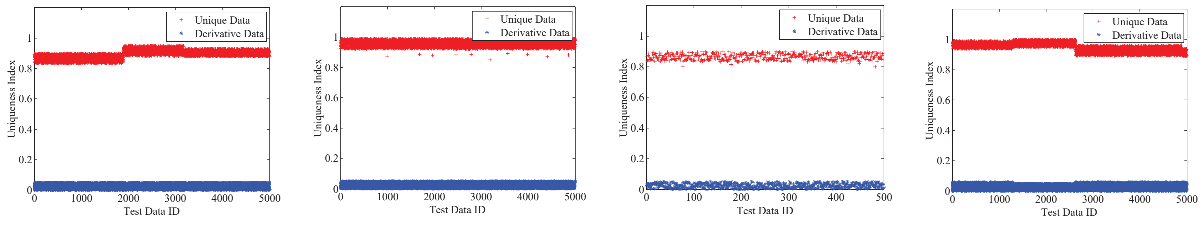
Uniqueness indices of text datasets and video datasets are lower than others for 'unique test data' group in general. This is because even after removing the stop words, text documents may have certain small amount of overlaps (*i.e.,* words are not listed in stop words list but are common) even if data is from different datasets. The results of text types and table types have several intervals because uniqueness index of text data slightly depends on the contents of datasets, and we chose test data from three datasets of different categories for text and table.

It is clear that the uniqueness indices can be clustered into two clusters with simple horizontal separators $y = \theta_{\mathsf{similar}}$ and $y = \theta_{\mathsf{unique}}$ where all unique data is above $y = \theta_{\mathsf{unique}}$ and the rest is below $y = \theta_{\mathsf{similar}}$. Throughout the simulation and emulation, we did not see any data that has low uniqueness index while it is in the ground truth set of unique data and vice versa. However, the actual values of the separators are dataset-dependent, and AccountTrade needs to find $\theta_{\mathsf{unique}}, \theta_{\mathsf{similar}}$



(a) Text per words # (b) Image per feat. # (c) Video per frame # (d) Table per row # (e) Table per column #

Figure 3. Benchmark of Uniqueness Index calculation



(a) Text (b) Image (c) Video (d) Table

Figure 4. Uniqueness index distribution of perturbed existing data v.s. new data

adaptively. We let AccountTrade first sets initial values for the thresholds (*e.g.,* $\theta_{\text{unique}} = 0.8, \theta_{\text{similar}} = 0.2$), and updates the thresholds when he pushed the examined data to manual inspection because the index is in $[\theta_{\text{similar}}, \theta_{\text{unique}}]$. If the manual inspection determines that it is derivative, the index value is assigned to θ_{similar} and vice versa. In case even the manual inspection does not know whether this data is unique, thresholds remain same. The distance between these two separators will strictly decrease every time gray-area data is determined as derivative or unique. Although we were not able to observe the dataset which makes two separators meet in the middle, we cannot conclude the distance will be large for all text, image, video, table datasets since the variety of our datasets is limited. Further exploring more datasets with variety is our future work. However, we conjecture that the distance of two separators will converge with a high probability for a given dataset that has similar characteristics. Then, brokers can adaptively explore and use different separators for different types of datasets (*e.g.,* different thresholds for SF novels, drama novels, history novel, *etc.*).

C. QoS from Emulation

We used our 10 COTS computers to concurrently generate data publication requests to AccountTrade deployed at our server computer. Our deployed program of AccountTrade responds to concurrent requests with multiple threads, and each request is processed with parallel algorithms. For each type, we measured the influence of concurrent requests per second to the latencies increased from the benchmark, which is indicative of QoS at the entities' side. The results are presented in Fig. 5, which present the minimum, average, and maximum increased latencies for each type. We plotted the results until all concurrent results are answered and terminated without exceptions. The server starts to reject table-type requests at 8 because of the large memory consumption (bloom filter's memory size is 2GB in our implementation). Texts' and images' requests are concurrently handled without noticeable QoS degradation for a while, and the degradation becomes prominent when the memory consumption rose up to 32GB and garbage collection occurs frequently. The video types' extra latencies are different from the previous two because the video loading is expensive, and the degradation is caused by the concurrent reading at the disk. Note that this QoS is bounded to our implementation in our hardware environment only. It will be improved if our server is equipped with more memory or more cores.

VI. Related Works

A. Accountability Systems

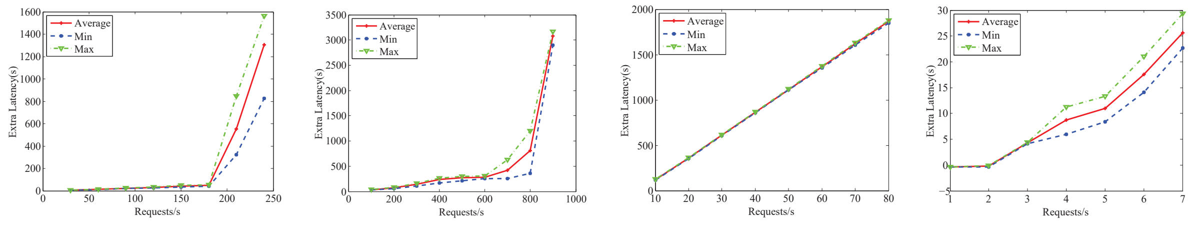
Accountability system has been studied in many other areas. Logging mechanisms are used to achieve accountability in Wireless Sensor Networks [39]; trusted IP manager is employed for accountability in internet protocol [32]; byzantine fault detection [21] is studied to account for faults in distributed systems; memory attestation protocol is proposed to achieve accountability against energy theft attackers in smart grids [34]; and finally, accountability in virtual machines [20] is studied to secure the cloud environment.

B. Copy-detection

Besides the fast variants we presented in §III-B, alternatives existing in the literature except for tables or graphs.

Text: Shingling along with MinHashing has long been used in the text copy detection [10] to discover similar text documents. W-shingling [11], shingling by words instead of letters, is also proposed to capture the word-based similarity. Charikar *et al.* proposed SimHash in [12] in order to detect near-duplicate text documents, and they also convert documents to high-dimensional vectors and small-size hash values. Approaches not based on shingling are also available.

Image: [14] introduces two approaches to perform copy detection: Locality Sensitive Hash on color histograms and MinHash on feature descriptors. In both approaches, the image is treated as a set of elements. In [26], feature descriptors are extracted from each image, and MinHash is applied to them, after which the Jaccard Index is approximated by the MinHash



(a) Text (b) Image (c) Video (d) Table

Figure 5. Emulated increased latency per requests/s

values. Geometric MinHash [13] is also proposed later, which is also compatible with AccountTrade.

Video: Video copy-detection is deeply related to that of image datasets. The major approach is to select keyframes and compare the similarities of the keyframes [17], [29].

VII. Conclusion

We proposed AccountTrade for big data trading among dishonest consumers. It guarantees correct book-keeping and achieves accountability by blaming dishonest consumers if they did not fulfill responsibilities in the transactions. Notably, to achieve the accountability against dishonest sellers who may re-sell others' datasets, we presented a novel rigorous measurement of the dataset uniqueness – uniqueness index – which can be efficiently computable. We formally defined the accountability model and proved it with ProVerif and rigorous analysis, and we also evaluated the performance and QoS using real-world datasets in our implemented testbed.

References

[1] Data markets compared – a look at data market offerings from four providers. goo.gl/k3qZsj.

[2] Ftc charges data broker with facilitating the theft of millions of dollars from consumers' accounts. goo.gl/7ygm7Q.

[3] Ftc charges data brokers with helping scammer take more than $7 million from consumers' accounts. goo.gl/kZMmXn.

[4] Ftc complaint offers lessons for data broker industry. goo.gl/csBYA3.

[5] Multimedia computing and computer vision lab. goo.gl/pbKeCj.

[6] R. Araújo, S. Foulle, and J. Traoré. A practical and secure coercion-resistant scheme for remote elections. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2008.

[7] D. Baltieri, R. Vezzani, and R. Cucchiara. Sarc3d: a new 3d body model for people tracking and re-identification. In *ICIAP*, pages 197–206. Springer, 2011.

[8] B. Blanchet. Automatic verification of security protocols in the symbolic model: The verifier proverif. In *FOSAD*, pages 54–87. Springer, 2014.

[9] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[10] A. Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*, pages 21–29. IEEE, 1997.

[11] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8):1157–1166, 1997.

[12] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388. ACM, 2002.

[13] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *CVPR*, pages 17–24. IEEE, 2009.

[14] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *CIVR*, pages 549–556. ACM, 2007.

[15] F. T. Commission et al. Data brokers: A call for transparency and accountability. 2014.

[16] S. A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158. ACM, 1971.

[17] M. Douze, H. Jégou, and C. Schmid. An image-based approach to video copy detection with spatio-temporal post-filtering. *Transactions on Multimedia*, 12(4):257–266, 2010.

[18] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *TON*, 8(3):281–293, 2000.

[19] F. Galasso, N. S. Nagaraja, T. Jimenez Cardenas, T. Brox, and B. Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *ICCV*, pages 3527–3534. IEEE, 2013.

[20] A. Haeberlen, P. Aditya, R. Rodrigues, and P. Druschel. Accountable virtual machines. In *OSDI*, pages 119–134, 2010.

[21] A. Haeberlen, P. Kouznetsov, and P. Druschel. Peerreview: Practical accountability for distributed systems. In *SIGOPS*, volume 41, pages 175–188. ACM, 2007.

[22] J. Heather and D. Lundin. The append-only web bulletin board. In *Formal Aspects in Security and Trust*, pages 242–256. Springer, 2008.

[23] K. Khurana and M. Chandak. Key frame extraction methodology for video annotation. *IJCEIT*, 4(2):221–228, 2013.

[24] J. Kim, D. Han, Y.-W. Tai, and J. Kim. Salient region detection via high-dimensional color transform and local spatial support. *IEEE Transactions on Image Processing*, 25(1):9–23, 2016.

[25] R. K. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee. Trustcloud: A framework for accountability and trust in cloud computing. In *SERVICES*, pages 584–588. IEEE, 2011.

[26] D. C. Lee, Q. Ke, and M. Isard. Partition min-hash for partial duplicate image discovery. In *ECCV*, pages 648–662. Springer, 2010.

[27] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. goo.gl/x5vRjJ.

[28] M. Lichman. UCI machine learning repository, 2013.

[29] Z. Liu, T. Liu, D. C. Gibbon, and B. Shahraray. Effective and scalable video copy detection. In *MIR*, pages 119–128. ACM, 2010.

[30] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[31] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[32] D. Naylor, M. K. Mukerjee, and P. Steenkiste. Balancing accountability and privacy in the network. *SIGCOMM*, 44(4):75–86, 2015.

[33] A. Rajaraman, J. D. Ullman, J. D. Ullman, and J. D. Ullman. *Mining of massive datasets*, volume 77. Cambridge University Press Cambridge, 2012.

[34] K. Song, D. Seo, H. Park, H. Lee, and A. Perrig. Omap: One-way memory attestation protocol for smart meters. In *ISPAW*, pages 111–118. IEEE, 2011.

[35] Z. Sun, H. Wang, H. Wang, B. Shao, and J. Li. Efficient subgraph matching on billion node graphs. *Proceedings of the VLDB Endowment*, 5(9):788–799, 2012.

[36] S. J. Swamidass and P. Baldi. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. *Journal of chemical information and modeling*, 47(3):952–964, 2007.

[37] T. Truderung, A. Vogt, et al. Accountability: definition and relationship to verifiability. In *CCS*, pages 526–535. ACM, 2010.

[38] D. Tsai, Y. Jing, Y. Liu, H. A.Rowley, S. Ioffe, and J. M.Rehg. Large-scale image annotation using visual synset. *ICCV*, 2011.

[39] Y. Xiao. Flow-net methodology for accountability in wireless networks. *Network, IEEE*, 23(5):30–37, 2009.

[40] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2015.

[41] L. Zhang, T. Jung, C. Liu, X. Ding, X.-Y. Li, and Y. Liu. Pop: Privacy-preserving outsourced photo sharing and searching for mobile devices. In *IEEE ICDCS*. IEEE, 2015.