AccountTrade: Accountability Against Dishonest Big Data Buyers and Sellers

Taeho Jung, *Member, IEEE*, Xiang-Yang Li, *Fellow, IEEE*, Wenchao Huang, Zhongying Qiao, Jianwei Qian, Linlin Chen, Junze Han, and Jiahui Hou

***Abstract*— In this paper, a set of accountable protocols denoted as AccountTrade is proposed for big data trading among dishonest consumers. For achieving a secure big data trading environment, AccountTrade achieves book-keeping ability and accountability against dishonest consumers throughout the trading (i.e., buying and selling) of datasets. We investigate the consumers' responsibilities in the dataset trading, then we design AccountTrade to achieve accountability against dishonest consumers that are likely to deviate from the responsibilities. Specifically, a *uniqueness index* is defined and proposed, which is a new rigorous measurement of the data uniqueness for this purpose. Furthermore, several accountable trading protocols are presented to enable data brokers to blame the misbehaving entities when misbehavior is detected. The accountability of AccountTrade is formally defined, proved, and evaluated by an automatic verification tool as well as extensive simulation with real-world datasets. Our evaluation shows that AccountTrade incurs at most 10-kB storage overhead per file, and it is capable of 8–1000 concurrent data upload requests per server.**

***Index Terms*— Data trading, accountability, fuzzy decuplication.**

I. Introduction

THE data trading industry has been increasing rapidly (*e.g.,* CitizenMe, DataExchange, Datacoup, Factual, Qlik, XOR Data Exchange, Terbine). These *data brokers* provide B2C or C2C dataset trading services, and they are the counterparts of physical commodities trading platforms, *e.g.,* Ebay and Amazon. Trading of digital datasets has become a trend as the trading of digital datasets became a promising business in the big data era [1]. Although profits lie in big data, organizations possessing large-scale datasets (companies or research institutes) do not participate in the data trading due to serious concerns in user-generated data [27], [28], [35], [52]. One of the major concerns is that we do not have accountability in the digital data trading [21], [25], [40]. The concerns are particularly huge due to the non-physical nature of the digital dataset – replication and delivery are almost costless when compared to physical commodities. Concerns arise at the broker side: data owners worry that brokers may illegally disclose or resell the datasets they outsourced to the brokers. On the other hand, concerns arise at the consumer side as well: dishonest consumers may illegally resell the purchased datasets. Addressing the first issue is possible because that was one of FTC's main concerns [16], and FTC has managed to detect and punish dishonest data brokers already [2]–[4]; and 2) achieving accountability in small-size systems has shown to be possible [21], [24]. Therefore, it is possible to monitor the broker's side. The second issue is hard to address. It is reasonable to monitor the broker's side (recent movement also shows that [16]), but it is hardly acceptable to monitor the consumers. Firstly, it is not lawful to lively monitor individual consumers' behavior because of the privacy implications. Secondly, the history of Internet suggests that any service requires installation of a heavy monitoring system cannot survive because it leads to bad user experiences.

We extend from our conference publication [29] and propose a suite of accountable protocols, denoted as AccountTrade, for big data trading hosted by data brokers. AccountTrade enables brokers to attain accountability against dishonest consumers throughout the trading by detecting their misbehavior. The trading-related misbehavior defined in this paper includes **tax evasion, denial of purchase**, and **resale of others' datasets**. Note that we do not try to detect idea-wise plagiarism (*e.g.,* novels with similar plots, images taken at the same scenery spot, videos taken with similar angles) because originality is a subjective factor that is hardly decidable even by human. Instead, we propose to detect the *blatant copy* (not an *exact* copy) in the datasets uploaded by owners, by detecting whether the given datasets are derived from others that have been uploaded before. Notably, the fuzzy copy-detection in table-type datasets or JSON-like datasets has not been studied yet to the best of our knowledge, and AccountTrade is the first to propose a feasible mechanism. The extra overhead incurred at the ordinary consumers' side is negligible when compared to the overhead of datasets uploading/downloading, and the extra overhead introduced at the brokers' side is also acceptable. We define a formal model for accountability which is proved and quantitatively analyzed. For the symbolic model, we use automatic proof with formal

Manuscript received November 20, 2017; revised March 28, 2018; accepted June 8, 2018. Date of publication June 18, 2018; date of current version July 23, 2018. This work was supported in part by the National Key R&D Program of China under Grant 2018YFB0803400, in part by the China National Funds for Distinguished Young Scientists under Grant 61625205, in part by the Key Research Program of Frontier Sciences, CAS, under Grant QYZDY-SSW-JSC002, in part by NSFC under Grant 61520106007, Grant 61751211, and Grant 61572453, and in part by NSF CNS under Grant 1526638. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Damien Vergnaud. *(Corresponding author: Xiang-Yang Li.)*

T. Jung and Z. Qiao are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556 USA.

X.-Y. Li and W. Huang are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: xiangyangli@ustc.edu.cn).

J. Qian, L. Chen, J. Han, and J. Hou are with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2018.2848657

language to prove the accountability, and for the computational model, we quantitatively analyze the accountability guarantee of AccountTrade.

Several challenges exist in designing AccountTrade. Firstly, the boundary for whether sale is legitimate or not is hard to define. There are two reasons for that: (1) dishonest sellers may bring perturbation into others' datasets before they try to resell them; defining how much overlap will make a dataset *copy* of another one is challenging; (2) data brokers buy and sell a huge number of large-scale datasets, but illegal resale monitoring inherently involves certain form of scanning over all datasets that they possess. Finally, as aforementioned, AccountTrade should impose negligible overhead to consumers and practically acceptable overhead to brokers. We have addressed these challenges by defining and proposing *uniqueness index* in AccountTrade, which is a new rigorous measurement of the data uniqueness. This index enables AccountTrade to detect dataset resale efficiently even in trading of big data.

We have the following contributions:

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

2. We efficiently detect illegal resale by defining and proposing the *uniqueness index* that is consistent with state-of-the-art data similarity comparison for different types. Notably, no such mechanism is available for table-type datasets, and existing mechanisms for JSON-like datasets are not scalable. Therefore, we present novel mechanisms to efficiently measure the similarity for those types.

3. AccountTrade is highly scalable with both number and volume of datasets. The extra overhead introduced at the users' side remains constant regardless of the data brokers' scale, and the extra overhead at the brokers' side grows linearly with the number of datasets only. The volumes of existing datasets do not contribute to the extra overhead.

II. Definitions and Models

A. Data Trading With Brokers

In our model, there are three entities: *brokers*, *sellers*, and *buyers*. Each entity has its own trading-related *responsibilities*.

1) Broker: Brokers provide shopping services in general (product listing, description, payment, delivery, *etc.*). Besides, they are in charge of book-keeping for accounting purposes (*i.e.,* recording trading transactions), and they also need to define what type of transaction is considered as reselling and should be prohibited.

2) Seller: Sellers are required to sell only the datasets that are collected/generated by themselves, and they should not resell others' datasets by slightly perturbing them. Also, they have to correctly file the tax report regarding the dataset transaction. They should not interrupt brokers' book-keeping.

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

Some areas are important but orthogonal to ours. Description of datasets' quality/utility for buyers [49] is complementary. Furthermore, we let sellers set the prices, but more sophisticated pricing mechanism may be considered. The accountability we study for data trading is independent from them.

B. Adversary Model & Channel Assumption

1) Malicious Users: Users may try to deviate from the responsibilities described above. Namely, they may *e.g.,* disrupt the brokers' data trading service, deny cleared transactions (*i.e.,* paid and sold) and resell previously purchased datasets. A user is defined as a *dishonest* user if he avoided any of the trading-related responsibilities, 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 copy detection mechanisms.

2) 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 [40], and the assumption that the brokers will be strictly supervised is also consistent with the FTC's recent action [16].

3) 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

The modeling in [46] is inherited to define a formal accountability model for AccountTrade. Our accountable protocols are characterized by 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 that all misbehaving entities must be blamed, is impossible to satisfy because "some misbehavior cannot be observed by any honest party" [46]. AccountTrade also requires *individual accountability*, which states that it must be able to correctly blame one or more parties unambiguously, rather than to blame a group without knowing the exact misbehaving person.

We define two formal models of accountability with different purposes. *symbolic individual accountability* is defined in a setting where all building blocks are abstracted as ideal black boxes. The symbolic model is amenable to automatic security verification protocols, *e.g.,* [8], who verify whether security flaws exist. Then, *computational individual accountability* without the abstraction is defined to give a quantitative analysis of individual accountability guarantee.

1) Symbolic Individual Accountability: A *verdict* is a boolean formula ψ which includes propositions having the form $\mathsf{dis}(e)$, where $\mathsf{dis}(e)$ is a statement "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. We use the expression $r \Rightarrow \psi$ to denote that ψ evaluates to true in

TABLE I
IMPORTANT SYMBOLS AND THEIR DEFINITIONS

Symbol	Term	Section to find its definition
H	Cryptographic Hash	§III-A, Page 3
$\mathsf{sig}_E(m)$	Signature	§III-A, Page 3
post_t	Post	§III-A, Page 3
$U_{\mathbb{D}}(d)$	Uniqueness Index	§III-C, Page 3
$\Delta(S_1, S_2)$	Similarity Function	§III-C, Page 4
$J(S_1, S_2)$	Jaccard Index	§III-C, Page 4
$mh_\pi(d)$	MinHash	§III-C, Page 4
M	# of MinHash Values	§III-C, Page 4
$B_m^k(d)$	Bloom Filter	§III-C, Page 5

the run r. Then, if a run r contains misbehavior(s) and ψ describes 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 '$\wedge$'.*

2) Computational Individual Accountability: The computational version is similar to the symbolic one except that we consider the 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

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. AccountTrade is composed of Upload, Examine, and Download. Please refer to Table I for notations.

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.

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

Examine: The broker examines whether the dataset is derived from existing ones with this protocol. He generates a set of *MinHash* values $mh_\pi(d)$ 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.

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

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

Φ is defined as $\{r \Rightarrow \mathsf{dis}(e) | r \in \Gamma\}$ where Γ is a set of runs which contain misbehavior. Our goal is to design the protocols such that Upload, Examine, and Download have both symbolic individual Φ-accountability and computational individual Φ, η, χ-accountability *w.r.t.* the broker.

III. SPECIFICATIONS OF ACCOUNTTRADE

A. Building Blocks

1) Cryptographic Hash: Suppose Σ is a set of characters. We employ 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.

2) 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.

3) Append-Only Bulletin Board: A bulletin board with 'append' and 'read' privileges only [26] 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.

With the building blocks, we present the architecture of AccountTrade as in Fig. 2. Buyers/sellers post posts at the bulletin board whenever they buy/sell datasets, and brokers verify the corresponding records exist before accepting/releasing datasets. After accepting a dataset, the brokers examine it before finally listing it for selling.

B. Upload for Sale

When a seller A wants to upload a dataset 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 sends the upload request along with

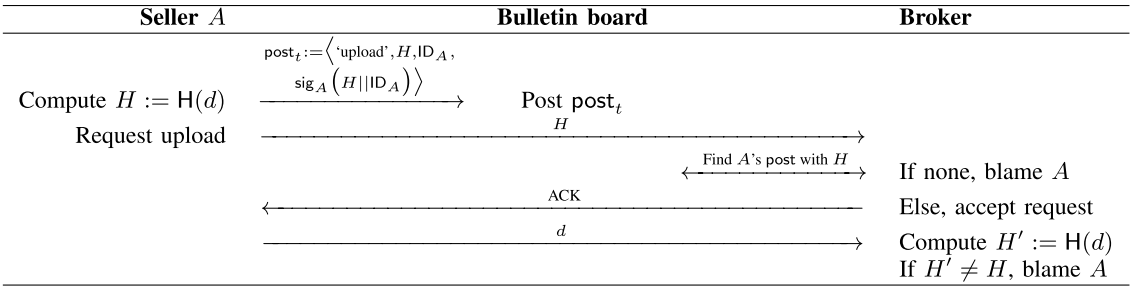


Fig. 1. Upload protocol between a seller A with ID ID_A and the broker for uploading dataset d.

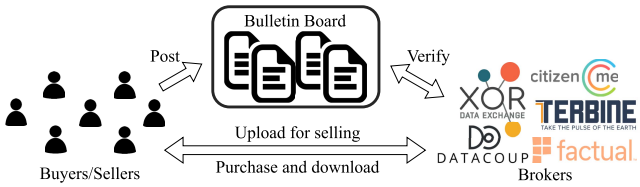


Fig. 2. Architecture of AccountTrade.

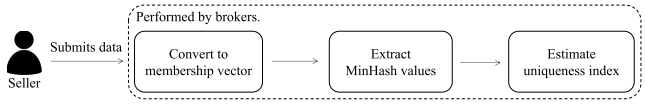


Fig. 3. Uniqueness index calculation performed by brokers.

$\mathsf{H}(d)$ to the broker. 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 and publishes the description of the dataset d (*e.g.,* its contents, price, $\mathsf{H}(d)$).

C. Dataset Examination

If the upload is successful, the broker checks whether a similar dataset has been uploaded before. To do so, we propose *uniqueness index*, which is indicative of the amount of overlaps between a given set S and 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:*

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

$J(S_1, S_2)$ denotes Jaccard Index, which is statistical measurement of the similarity 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, d is the dataset to be examined, 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 define and use this uniqueness index in dataset re-selling detection is manifold. Firstly, it intuitively measures how many elements of S_x are similar to the elements in the entire set $\mathbb{S}$, and the multiplier after the 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 § VII). Thirdly, there is no known similarity comparison mechanism for table-type datasets, and similarity comparison of JSON-like datasets are hardly scalable. We were able to design a novel similarity mechanism with high scalability based on this uniqueness index. Fourthly, our extensive experiments on the real-world datasets (§ V-B) show that the uniqueness index of d_x is low (≤ 0.2) 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 filters many illegal reselling requests automatically with a high confidence when the threshold values are selected conservatively. We will elaborate how to choose the thresholds in § V-B. Finally, because uniqueness indices are clearly separated, it becomes possible to rely on threshold-based decisions which is efficient, and this enhances the scalability.

Next, we describe how to calculate the aforementioned uniqueness index. The flow is sketched in Fig. 3.

For a given dataset d, we first convert it to a *membership vector* which uses a binary vector to represent d. Then, we calculate the MinHash values of the membership vector, which will be used to estimate the *uniqueness index* (Def. 3).

To have a better insight into why we represent datasets with sets of elements, we briefly explain MinHash and its property.

1) MinHash: MinHash [11] 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: the permutation π suggests 1 in 7th bit, 2 in 4th bit, 3 in 3rd bit, 4 in 8th bit, and so forth, and if one probes the membership vector $\vec{\mathbf{S}}$ by following this order, the 8th bit is the first 1, therefore the MinHash value $mh_\pi(\vec{\mathbf{S}})$ is 8.

The following property from [12] 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, if we pre-define M permutations $\{\pi_1, \pi_2, \cdots, \pi_M\}$ and achieve M pairs of MinHash values from $\vec{\mathbf{S}}_1$ and $\vec{\mathbf{S}}_2$, *i.e.,* $\{(mh_{\pi_i}(\vec{\mathbf{S}}_1), mh_{\pi_i}(\vec{\mathbf{S}}_2)) | \pi_i, i = 1, 2, \cdots M\}$, we can use these pairs to approximate $J(\vec{\mathbf{d}}_1, \vec{\mathbf{d}}_2)$ by calculating the proportion of equal pairs out of M pairs. Such an approximation has a bounded expected error $O(1/\sqrt{M})$ according to the Chernoff-Hoeffding bound.

In other words, if datasets can be converted to sets of elements, and if enough permutations are defined to achieve MinHash values, one can easily approximate the uniqueness index of a dataset d against a database of datasets $\mathbb{D}$ by comparing the MinHash values. Only the conversion from datasets to sets of data elements has a complexity linear to the size of datasets, and the subsequent procedures' complexities are linear to M. Therefore, if we pre-define and re-use M permutations and store all calculated MinHash values, the uniqueness calculation's complexity does not depend on the sizes of the datasets that are already in the database.

To do so, we need define 'data elements' for different types of datasets. Any definition is accepted in AccountTrade, and we employ existing definitions for texts and images in Information Retrieval (IR) community as building blocks [10], [11], [32]. However, such definitions for table/JSON/XML do not exist in the literature yet, because retrieval of table/JSON/XML has not been studied much – retrieval algorithms for those types do not have wide applications in web unlike text retrieval or image retrieval. For those types, we present our own definitions which lead to novel similarity comparisons. Graphs are treated differently because the comparison involves an NP-complete problem.

2) Text: The text document is *shingled* to k-shingles (also known as k-grams) [12]. 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 as 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 [34]. 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 [39]). Finally, the MinHash values can be extracted from the vectors.

3) Image: Feature descriptors (*e.g.,* SIFT [50], [51]) are useful in object recognition [31]. 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.,* $\{0, 1\}^{dim}$, or real, *i.e.,* $[0, 1]^{dim}$. Since dim is as large as 128, it is impossible to directly turn the list of features to a membership vector in either case. 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.

4) Video: Keyframes extraction [30] 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).

5) 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 [20]. For the sake of simplicity, we use $B_m^k(d)$ 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 $B_m^k(d)$. Then, due to the bloom filters' properties, $B_m^k(d_1 \cup d_2) = B_m^k(d_1) \vee B_m^k(d_2)$ and $B_m^k(d_1 \cap d_2) = B_m^k(d_1) \wedge B_m^k(d_2)$, 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 $B_m^k(d)$ is $f(|B_m^k(d)|) = -\frac{m \ln(1 - |B_m^k(d)|/m)}{k}$ where $|B_m^k(d)|$ denotes the number of 1's in $B_m^k(d)$ [44], the Jaccard Index of d_1, d_2 is approximated by

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

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 $2^c - 1$ sub-tuples for every row into the bloom filter. Because of the performance concern in the relational database management (*e.g.,* owing to join operations and 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 has excessive columns,

we insert the tuples of size $1, 2, 3, \cdots, g$ only instead of inserting all sub-tuples. Then, the number of elements in the bloom filter is reduced from $O(2^c r)$ to $O(c^g r)$ at the cost of accuracy loss, where r is the number of rows in the table. 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 does not directly indicate 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 $\frac{|B_m^k(d_1) \wedge B_m^k(d_2)|}{|B_m^k(d_1) \vee B_m^k(d_2)|}$ (denoted as J_{12}) with MinHash values of $B_m^k(d_1)$ and $B_m^k(d_2)$. Therefore, if we can calculate $|B_m^k(d_1) \wedge B_m^k(d_2)|$ or $|B_m^k(d_1) \vee B_m^k(d_2)|$ from J_{12}, we are able to estimate $J(d_1, d_2)$ based on the bloom filters. We derive them by using the inclusion-exclusion principle:

$$|B_m^k(d_1) \vee B_m^k(d_2)| = \frac{|B_m^k(d_1)| + |B_m^k(d_2)|}{1 + J_{12}}$$

$$|B_m^k(d_1) \wedge B_m^k(d_2)| = J_{12} \cdot \frac{|B_m^k(d_1)| + |B_m^k(d_2)|}{1 + J_{12}}$$

To calculate them in a constant time, we augment the bloom filter and use an extra field to store $|B_m^k(d)|$ for each filter so that this field can be accessed in a constant time. Then, AccountTrade can approximately calculate the Jaccard Index $J(d_1, d_2)$ in a constant time irrelevant to the size of the filter.

6) Graph: If graphs are unlabeled, *i.e.,* nodes and edges do not have any attribute, the similarity comparison between two graphs only compares the graph topology. The most efficient similarity comparison for unlabled graphs is [43] which generates 6 GBytes index and 33s index time for a graph of 1 billion nodes. However, even in [43], the query time is several seconds for a graph with only tens of nodes and edges. This is due to the inherent hardness of the sub-graph isomorphism problem which is NP-complete [17]. 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 graphs as tables and use the same method as aforementioned.

7) JSON/XML: XML files can be easily converted to JSON format and vice versa, therefore we only discuss how to handle JSON files. They are essentially text files with a (key:value) pair-like dictionary structure, but they cannot be treated as texts or tables. Their structure is closer to trees. The top level can be one or multiple (key:value) pairs which are root node(s) of the trees, and a value in one (key:value) pair can be another (key:value) pair nested in it, which is a child of the node. In JSON-like datasets, the topology among the (key:value) pairs (*i.e.,* the tree structure) contains rich information, and the similarity comparison needs to capture it. To do so, we treat every subtree in the tree/forest as an element in a JSON file. We apply a cryptographic hash in order to map every subtree to a finite domain, which is considered as the domain for the membership vector.

Although the domain is finite and discrete, it is impractical to use the previous way to calculate the MinHash values. The number of elements in JSON files is usually less than 10^6 while the size of the domain is as large as $2^{256} \approx 10^{77}$. This implies we must probe $\approx 10^{77}$ times on average in a sparse membership vector. Instead, for every element in a JSON file, we apply a hash function H which maps it to a positive integer value. Then, from all the integers generated from all the elements, we find the minimum integer and set it as the MinHash value of the file, denoted as $mh_H(d)$. Now, the MinHash is calculated based on H rather than a permutation π. If the same hash function H is used to compute such a MinHash values for two sets S_1, S_2, we still have $\Pr[mh_H(\vec{\mathbf{S}}_1) = mh_H(\vec{\mathbf{S}}_2)] = \|\vec{\mathbf{S}}_1 \wedge \vec{\mathbf{S}}_2\| / \|\vec{\mathbf{S}}_1 \vee \vec{\mathbf{S}}_2\|$ for any two sets S_1, S_2 [34], where $mh_H(\vec{\mathbf{S}})$ refers to the MinHash value of the set S calculated with the hash function H. The complexity of this MinHash is linear *w.r.t.* the number of elements in the JSON file. Then, instead of having M permutations, we can have M different hash functions to generate M MinHash values for JSON files.

Because every subtree is considered as an element, as long as the attackers' perturbation does not damage each element, the uniqueness index will capture the similarity among JSON files. Similar to the table, there are exponentially many subtrees in a tree, and we limited the depth of the subtrees within a constant for getting a polynomial-time computation complexity.

8) For All Types: After the uploaded dataset d's uniqueness index is calculated with a linear scan on $\mathbb{D}$, the broker can decide whether he will accept it (if the uniqueness index $U_{\mathbb{D}}(d)$ is very high), reject it (if $U_{\mathbb{D}}(d)$ is very low), or leave it to manual inspection (otherwise). How to set up the thresholds will be explained in § V-B.

D. 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. 4). 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.

E. Parallelization

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

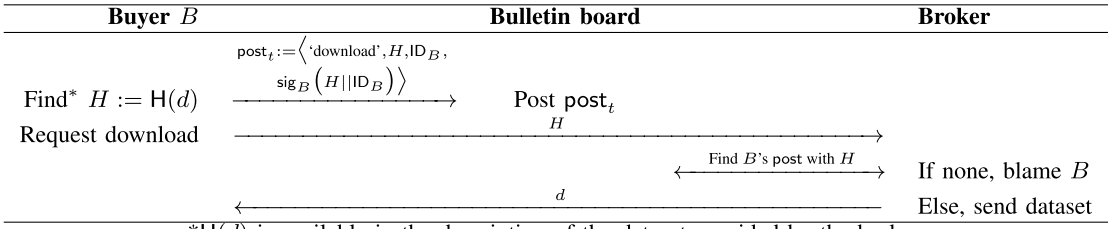


Fig. 4. Download protocol between a buyer B with ID ID_B and the broker for downloading dataset d.

MinHash values. Therefore, we consider reducing the overall execution time by introducing parallelization. Data is logically partitioned into chunks (except JSON/XML types) 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.

1) Text: We divide text documents 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.

2) Image: We divide images 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.

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

4) Table & Graph: We horizontally partition tables into several chunks sub-tables. Each processor can independently load each sub-tuple to the shared bloom filter simultaneously.

5) JSON/XML: JSON files are hard to divide because the tree structure needs to be preserved. Therefore, we do not parallelize the I/O, and we only parallelize the MinHash calculation.

F. Accountability Properties of AccountTrade

Upload

1) 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.

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

Examine

3) 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

4) 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 modeling 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 modeled 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 enumerate all misbehavior. Then, we add events for the broker to blame dishonest participants when it detects them. 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 https://goo.gl/TNfF8n, 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

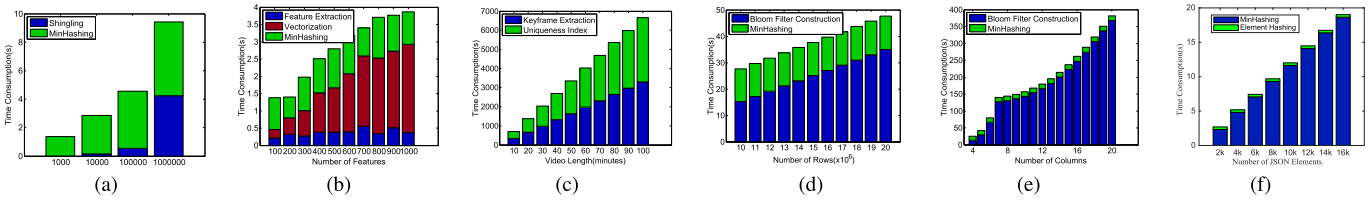


Fig. 5. Benchmark of Uniqueness Index calculation. (a) Text per words #. (b) Image per feat. #. (c) Video per frame #. (d) Table per row #. (e) Table per column #. (f) JSON per element #.

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, $\eta_{\mathrm{J2}} = \eta_{\mathrm{J4}} = 0$. 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 $\eta_{\mathrm{J3}} = 1 - (1 - \delta_{mh})(1 - \delta_{\mathsf{conv}})$. 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, $\eta = \max(\eta_{\mathrm{J1}}, \cdots, \eta_{\mathrm{J4}}) = \eta_{\mathrm{J3}}$.

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 bookkept. However, the probability of this is as small as $\frac{1}{2^\kappa}$ where κ is the security parameter, therefore $\chi_{\mathrm{J1}} = \frac{1}{2^\kappa}$ where $\chi_{\mathrm{J1}} = \Pr[\neg(\text{Broker ensures } \Phi)]$ for a run with dishonest seller in **J1**. Similarly, $\chi_{\mathrm{J4}} = \chi_{\mathrm{J2}} = \frac{1}{2^\kappa}$. Similar analysis applied to χ_{J3} shows that $\chi_{\mathrm{J3}} = \eta_{\mathrm{J3}} = \delta_{mh} + \delta_{\mathsf{conv}} - \delta_{mh}\delta_{\mathsf{conv}}$. Hence, $\chi = \max(\frac{1}{2^\kappa}, \delta_{mh} + \delta_{\mathsf{conv}} - \delta_{mh}\delta_{\mathsf{conv}})$. ■

V. Validation on Prototype Testbed

We acquired texts, images, tables, graphs and JSON datasets from publicly available sources. Text datasets are from [33] and [37]; image datasets are from [5] and [47]; video datasets are from [7] and [22]; table & graph datasets are from [33] and [37]; and JSON datasets are from US Government Open Data site [23]. Unlike other data types, we were not able to acquire enough number of video or JSON files from the publicly available datasets. The total volume is 2.2 TBytes, and they are used in the following benchmark, simulation, and the emulation.

A prototype system of AccountTrade is deployed in a server with Intel(R) Xeon E5-2620 and 32GB DDR4 1866 for emulation using the first four types, and the extension to JSON-like datasets is deployed in another server with Intel(R) Xeon E5-2680 and 256GB RAM [1]

A. Microbenchmark With Real Data

Note that we implemented the parallel algorithms (§ III-E).

1) Extra Overhead of Upload *and* Download*:* The only extra information AccountTrade requires brokers to store are 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. 5. Fig. 5(a)-(e) present the time consumption in calculating 1024 MinHash values from different types of data. All the run time presented in Fig. 5 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 sizes as well. In Fig. 5(b), the time for feature extraction depends on the image resolution and the contents instead of the number of features, however the majority of total time (for converting to membership vectors) grows linearly *w.r.t.* the number of features since a nearest-neighbor search is involved for each feature descriptor. The uniqueness index calculation in the video type data (Fig. 5(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

[1]The lead author changed the institution recently, and he no longer had access to the previous computing environment.

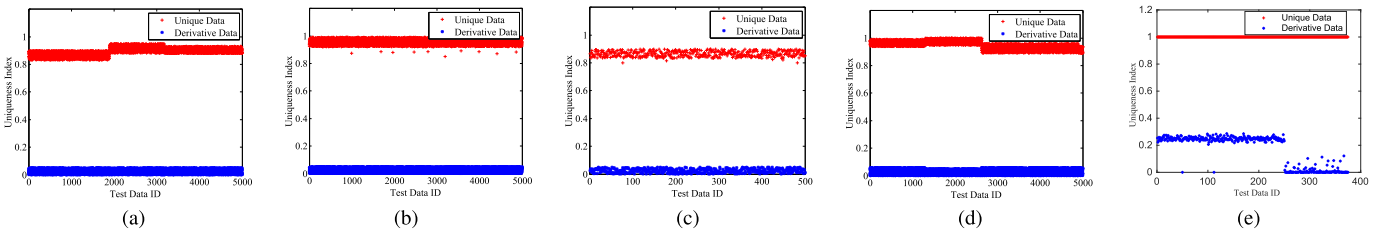


Fig. 6. Uniqueness index distribution of perturbed existing data v.s. new data. (a) Text. (b) Image. (c) Video. (d) Table. (e) JSON.

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. 5(e). Unlike other types, the time for MinHashing grows linearly *w.r.t.* the data sizes in JSON (Fig. 5(f)) because we calculate MinHash differently.

Examine is executed when sellers upload their dataset, and it does not have to be performed in a real time since it is acceptable that a hold is placed on an uploaded dataset when it is uploaded for selling. 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 and JSON types who contain MinHash values of 500/375 files respectively. Then, we acquired a ground truth datasets of *unique data* and *derivative data* as follows. We considered datasets A and B are unique from each other if they are from different sources, and we considered A as derivative if it is achieved by applying these perturbations on B:

- Text: Replace words with synonyms; switch active/passive tenses of verbs; delete sentences; merge documents.
- Image & video: Crop the image/video; rotate the image/video; merge existing images/video.
- Table & graph: Delete rows; delete columns.
- JSON/XML: Remove nodes; disorder nodes; extract nodes in trees/forests.

The consequent distributions are shown in Fig. 6.

Uniqueness indices of text datasets and video datasets (Fig. 6(a), Fig. 6(c)) 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 documents are from different datasets. On the other hands, JSON datasets' uniqueness indices (Fig. 6(e)) are close to 1 for 'unique test data' because SHA-256 is applied to map each subtree to an element. The domain of the digest is as large as 2^{256}, and hash collision did not occur in our experiment. The results of text types and table types have several intervals (Fig. 6(a), Fig. 6(d)) 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. Some perturbed JSON files' uniqueness indices (Fig. 6(e)) are as high as 0.2 because the hashed subtree changes if some of its children are removed, but the similarity can still be captured from other common parts.

It is clear that the uniqueness indices can be clustered into two clusters with simple horizontal separators $y = \theta_{\mathsf{low}}$ and $y = \theta_{\mathsf{high}}$ where all unique data are above $y = \theta_{\mathsf{high}}$ and the rest is below $y = \theta_{\mathsf{low}}$. 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 $\theta_{\mathsf{high}}, \theta_{\mathsf{low}}$ adaptively. We let AccountTrade first sets initial values for the thresholds (*e.g.,* $\theta_{\mathsf{high}} = 0.8, \theta_{\mathsf{low}} = 0.3$), and updates the thresholds when the index falls into $[\theta_{\mathsf{low}}, \theta_{\mathsf{high}}]$. If the manual inspection determines that it is derivative, θ_{low} is set to be that index value; if the manual inspection determines that it is unique, θ_{high} is set to be that index value. In case even the manual inspection does not know whether this file unique, thresholds remain same. The distance between these two separators will strictly decrease every time gray-area datasets are 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. 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 novels with different genres).

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 extra latencies caused by concurrent requests, which is indicative of QoS at the users' side. The results are presented in Fig. 7, which present the minimum, average, and maximum extra latencies for each type. We plotted the results until all concurrent results are answered and terminated without exceptions. The server starts

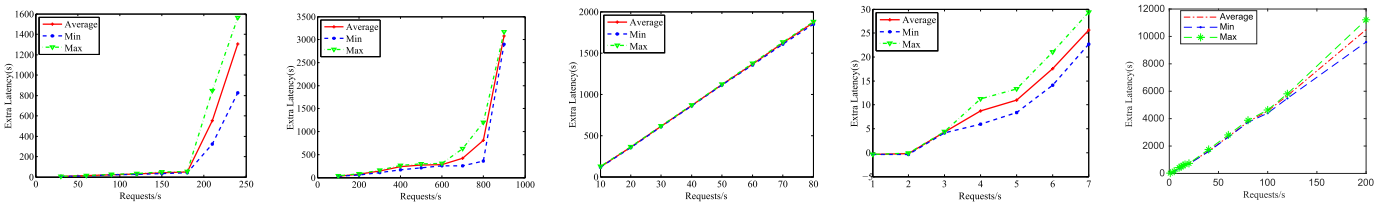


Fig. 7. Emulated increased latency per requests/s. (a) Text. (b) Image. (c) Video. (d) Table. (e) JSON.

to reject table-type requests at 8 requests (Fig. 7(d)) because of the large memory consumption (bloom filter's memory size is 2GB in our implementation). Texts' and images' requests (Fig. 7(a), Fig. 7(b)) 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 (Fig. 7(c)) because the video loading is expensive, and the degradation is caused by the concurrent reading at the disk. Because the examination of JSON takes more time than other types(Fig. 7(e)), it was not practical to increase the concurrency until we observe the bottleneck (which would take up to 10-15 days for one-time emulation if we had). However, we can still conjecture from the figure that there would be a bottleneck at around 300-400 requests/s. Note that this QoS is bounded to our hardware environment only. It will be improved if our server is equipped with more memory or more cores.

VI. Limitations and Discussion

A. Threshold-Based Decision

The detection mechanism involves copy-detection in various types of datasets, and we use the uniqueness index along with a threshold-based detection. We adopt the current threshold-based decision process because (1) it is extremely fast to make decisions with it, which makes it suitable for large-scale platforms where more than billions of datasets may reside; and (2) our extensive simulation with real-world datasets show that the uniqueness indices are clearly separable into two groups: unique and derivative datasets. In theory, it is possible to project datasets to another complex feature space so that more sophisticated perturbation can be captured (*e.g.,* using recent advancement such as deep neural networks), but a separate line of study needs to be performed to increase the efficiency, and the "copy" needs to be re-defined accordingly.

B. Plagiarism Detection

Determining whether a dataset is *original* or not is not within the scope of this paper as discussed in the introduction. Therefore, AccountTrade cannot detect idea-wise plagiarism.

VII. Related Works

This paper and its conference version [29] are the first to study consumer accountability in data trading. We first review recent works in data trading, accountability systems, and copy-detection Then, we compare this paper with our own conference version.

A. Data Trading

Data trading has become a popular research topic in recent years. Cao *et al.* [13] designed a data trading platform utilizing the iterative auction mechanism to achieve maximum efficiency to offset the self-interest behaviors amongst the players (data collectors, users and owners). The platform was created for multiple players who may compete with each other. Delgado-Segura *et al.* [18] proposed an innovative protocol that ensures fairness between data buyers and sellers so that neither party is required to trust the other during the whole transaction, as fairness is enforced by the said protocol based on the Bitcoin language. The outcome of the fair protocol is that payment and goods (data) delivery occur at the same time during each transaction.

B. Accountability Systems

Accountability system has been studied in many other areas. Logging mechanisms are used to achieve accountability in Wireless Sensor Networks [48]; trusted IP manager is employed for accountability in internet protocol [40]; byzantine fault detection [25] is studied to account for faults in distributed systems; memory attestation protocol is proposed to achieve accountability against energy theft attackers in smart grids [42]; and finally, accountability in virtual machines [24] is studied to secure the cloud environment.

C. Copy Detection

Besides the fast variants we presented in § III-C, alternatives exist except table/graph datasets.

1) Text: Shingling along with MinHashing has long been used in the text copy detection [11] to discover similar text documents. W-shingling [12], shingling by words instead of letters, is also proposed to capture the word-based similarity. Charikar proposed SimHash in [14] in order to detect near-duplicate text documents, and they also convert documents to high-dimensional vectors and small-size hash values.

2) Image: Chum *et al.* [15] introduce 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 [32], feature descriptors are extracted from each image, and MinHash is applied to them, after which the Jaccard Index is approximated by the MinHash values.

3) 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 [19], [38].

4) JSON/XML: XML similarity comparison has been surveyed in [45]. It includes a dynamic programming method based on tree edit distance [36] and a information retrieval method for document-centric XMLs [45]. For JSON types, a FOSS JavaScript library [41] enables calculation of similarities using a point-based mechanism. However, none of them is practical in our scenario where a broker needs to handle billions of datasets.

D. AccountTrade [29]

In [29], four types of common datasets (text, image, video, table) in dataset trading were discussed. XML/JSON-like datasets with (key:value) structurers were another common type which was not handled in the conference version. Because (key:value) pairs in XML or JSON datasets present a structure similar to a tree or a forest, the data examination mechanism needs to capture the similarity of the structure rather than the string values. In this paper, we present a similarity comparison mechanism for JSON datasets which is scalable enough for handling millions of datasets. The mechanism still uses the uniqueness index in the conference version. Therefore, AccountTrade in the extended version is a coherent framework which can handle the trading of text, image, video, table, and XML/JSON datasets.

VIII. Conclusion

This paper presents AccountTrade which guarantees correct book-keeping and achieves accountability in the big data trading among dishonest consumers. AccountTrade blames dishonest consumers if they deviate from their responsibilities in data transactions. To achieve accountability against dishonest sellers who may resell others' datasets, we presented a novel rigorous quantification of the dataset uniqueness – uniqueness index – which is efficiently computable. We formally defined two accountability models and proved them with ProVerif and theoretic analysis, and we also evaluated the performance and QoS using real-world datasets in our implemented testbed.

Acknowledgement

The authors would like to thank Cheng Su (School of Computer Science and Technology, University of Science and Technology of China) for implementing their ProVerif model.

References

[1] *Data Markets Compared—A Look at Data Market Offerings from Four Providers*. Accessed: Mar. 15, 2015. [Online]. Available: goo.gl/k3qZsj

[2] *FTC Charges Data Broker With Facilitating the Theft of Millions of Dollars from Consumers' Accounts*. Accessed: Mar. 15, 2015. [Online]. Available: goo.gl/7ygm7Q

[3] *FTC Charges Data Brokers with Helping Scammer Take More Than $7 Million from Consumers' Accounts*. Accessed: Mar. 15, 2015. [Online]. Available: goo.gl/kZMmXn

[4] *FTC Complaint Offers Lessons for Data Broker Industry*. Accessed: Mar. 15, 2015. [Online]. Available: goo.gl/csBYA3

[5] *Multimedia Computing and Computer Vision Lab*. Accessed: May 10, 2015. [Online]. Available: goo.gl/pbKeCj

[6] R. Araújo, S. Foulle, and J. Traoré, "A practical and secure coercion-resistant scheme for remote elections," in *Proc. Dagstuhl Seminar*, 2008. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-12980-3_20

[7] D. Baltieri, R. Vezzani, and R. Cucchiara, "SARC$_3$D: A new 3D body model for people tracking and re-identification," in *Proc. ICIAP*, Springer, 2011, pp. 197–206. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-24085-0_21

[8] B. Blanchet, "Automatic verification of security protocols in the symbolic model: The verifier proverif," in *Proc. FOSAD*, Springer, 2014, pp. 54–87. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-10082-1_3

[9] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[10] S. Brin, J. Davis, and H. Garcia-Molina, "Copy detection mechanisms for digital documents," in *Proc. SIGMOD*, vol. 24, 1995, pp. 398–409.

[11] A. Z. Broder, "On the resemblance and containment of documents," in *Proc. Compression Complexity Sequences*, Jun. 1997, pp. 21–29.

[12] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic clustering of the Web," *Comput. Netw. ISDN Syst.*, vol. 29, nos. 8–13, pp. 1157–1166, 1997.

[13] X. Cao, Y. Chen, and K. J. R. Liu, "An iterative auction mechanism for data trading," in *Proc. ICASSP*, Mar. 2017, pp. 5850–5854.

[14] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proc. STOC*, 2002, pp. 380–388.

[15] O. Chum, J. Philbin, M. Isard, and A. Zisserman, "Scalable near identical image and shot detection," in *Proc. CIVR*, 2007, pp. 549–556.

[16] *Data Brokers: A Call for Transparency and Accountability*, Federal Trade Commission, Washington, DC, USA, 2014.

[17] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. STOC*, 1971, pp. 151–158.

[18] D. G. Roy, B. Mahato, D. De, and R. Buyya, "Application-aware end-to-end delay and message loss estimation in Internet of Things (IoT)—MQTT-SN protocols," *Future Gener. Comput. Syst.*, to be published. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17329990, doi: 10.1016/j.future.2018.06.040

[19] M. Douze, H. Jégou, and C. Schmid, "An image-based approach to video copy detection with spatio-temporal post-filtering," *IEEE Trans. Multimedia*, vol. 12, no. 4, pp. 257–266, Jun. 2010.

[20] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: A scalable wide-area Web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, Jun. 2000.

[21] M. Felici, T. Koulouris, and S. Pearson, "Accountability for data governance in cloud ecosystems," in *Proc. CloudCom*, vol. 2, Dec. 2013, pp. 327–332.

[22] F. Galasso, N. S. Nagaraja, T. J. Cardenas, T. Brox, and B. Schiele, "A unified video segmentation benchmark: Annotation, metrics and analysis," in *Proc. ICCV*, Dec. 2013, pp. 3527–3534.

[23] U.S. Government. (2017). *The Home to U.S. Government's Open Data*. [Online]. Available: https://data.gov

[24] A. Haeberlen, P. Aditya, R. Rodrigues, and P. Druschel, "Accountable virtual machines," in *Proc. OSDI*, 2010, pp. 119–134.

[25] A. Haeberlen, P. Kouznetsov, and P. Druschel, "PeerReview: Practical accountability for distributed systems," *ACM SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 175–188, 2007.

[26] J. Heather and D. Lundin, "The append-only Web bulletin board," in *Formal Aspects in Security and Trust*. Springer, 2008, pp. 242–256. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-01465-9_16

[27] T. Jung, J. Han, and X.-Y. Li, "PDA: Semantically secure time-series data analytics with dynamic user groups," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 2, pp. 260–274, Mar./Apr. 2018.

[28] T. Jung, X.-Y. Li, and M. Wan, "Collusion-tolerable privacy-preserving sum and product calculation without secure channel," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 1, pp. 45–57, Jan. 2015.

[29] T. Jung *et al.*, "Accounttrade: Accountable protocols for big data trading against dishonest consumers," in *Proc. INFOCOM*, May 2017, pp. 1–9.

[30] K. Khurana and M. B. Chandak, "Key frame extraction methodology for video annotation," *Int. J. Comput. Aided Eng. Technol.*, vol. 4, no. 2, pp. 221–228, 2013.

[31] J. Kim, D. Han, Y.-W. Tai, and J. Kim, "Salient region detection via high-dimensional color transform and local spatial support," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 9–23, Jan. 2016.

[32] D. C. Lee, Q. Ke, and M. Isard, "Partition min-hash for partial duplicate image discovery," in *Proc. ECCV*, Springer, 2010, pp. 648–662. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-15549-9_47

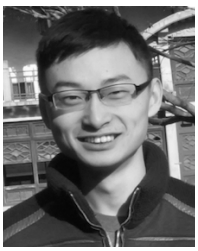
[33] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford Large Network Dataset Collection*. Accessed: Jan. 27, 2015. [Online]. Available: goo.gl/x5vRjJ
[34] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[35] X.-Y. Li, C. Zhang, T. Jung, J. Qian, and L. Chen, "Graph-based privacy-preserving data publication," in *Proc. INFOCOM*, Apr. 2016, pp. 1–9.
[36] W. Lian, D. W. L. Cheung, N. Mamoulis, and S.-M. Yiu, "An efficient and scalable algorithm for clustering XML documents by structure," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 1, pp. 82–96, Jan. 2004.
[37] M. Lichman. (2013). *UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/ml/index.php
[38] Z. Liu, T. Liu, D. C. Gibbon, and B. Shahraray, "Effective and scalable video copy detection," in *Proc. MIR*, 2010, pp. 119–128.
[39] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[40] D. Naylor, M. K. Mukerjee, and P. Steenkiste, "Balancing accountability and privacy in the network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 75–86, 2015.
[41] N. L. of Finland, (2017). *Json-Similarity*. [Online]. Available: https://github.com/NatLibFi/json-similarity
[42] K. Song, D. Seo, H. Park, H. Lee, and A. Perrig, "OMAP: One-way memory attestation protocol for smart meters," in *Proc. ISPAW*, May 2011, pp. 111–118.
[43] Z. Sun, H. Wang, H. Wang, B. Shao, and J. Li, "Efficient subgraph matching on billion node graphs," *VLDB Endowment*, vol. 5, no. 9, pp. 788–799, 2012.
[44] S. J. Swamidass and P. Baldi, "Mathematical correction for fingerprint similarity measures to improve chemical retrieval," *J. Chem. Inf. Model.*, vol. 47, no. 3, pp. 952–964, 2007.
[45] J. Tekli, R. Chbeir, and K. Yetongnon, "An overview on XML similarity: Background, current trends and future directions," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 151–173, 2009.
[46] R. Küsters, R. Küsters, and A. Vogt, "Accountability: Definition and relationship to verifiability," in *Proc. CCS*, 2010, pp. 526–535.
[47] D. Tsai, Y. Jing, Y. Liu, H. A. Rowley, S. Ioffe, and J. M. Rehg, "Large-scale image annotation using visual synset," in *Proc. ICCV*, Nov. 2011, pp. 611–618.
[48] Y. Xiao, "Flow-net methodology for accountability in wireless networks," *IEEE Netw.*, vol. 23, no. 5, pp. 30–37, Sep. 2009.
[49] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer, "Quality assessment for Linked data: A survey," *Semantic Web*, vol. 7, no. 1, pp. 63–93, 2015.
[50] L. Zhang, T. Jung, P. Feng, K. Liu, X.-Y. Li, and Y. Liu, "PIC: Enable large-scale privacy preserving content-based image search on cloud," in *Proc. ICPP*, 2015, pp. 949–958.
[51] 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 *Proc. IEEE ICDCS*, Jun./Jul. 2015, pp. 308–317.
[52] L. Zhang, X.-Y. Li, K. Liu, T. Jung, and Y. Liu, "Message in a sealed bottle: Privacy preserving friending in mobile social networks," *IEEE Trans. Mobile Comput.*, vol. 14, no. 9, pp. 1888–1902, Sep. 2015.

Taeho Jung received the B.E. degree from Tsinghua University in 2011 and the Ph.D. degree from the Illinois Institute of Technology in 2017. He is currently an Assistant Professor of computer science and engineering with the University of Notre Dame. His research areas include data security, user privacy, and applied cryptography. He has received the Best Paper Award from the IEEE IPCCC 2014. Two of his papers were selected as the Best Paper Candidate at the ACM MobiHoc 2014 and as the Best Paper Award Runner Up at the BigCom 2015.

Xiang-Yang Li (F'15) received the bachelor's degree from the Department of Computer Science, Tsinghua University, in 1995, the bachelor's degree from the Department of Business Management, Tsinghua University, in 1995, and the Ph.D. degree from the University of Illinois at Urbana–Champaign. He was a Full Professor with the Illinois Institute of Technology, Chicago, USA. He is currently a Full Professor and the Executive Dean of the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. He is an ACM Distinguished Scientist.

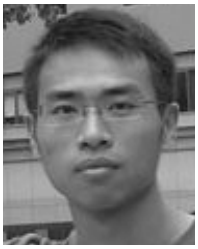
Wenchao Huang received the B.S. and Ph.D. degrees in computer science from the University of Science and Technology of China in 2005 and 2011, respectively. He is currently an Associate Professor with the School of Computer Science and Technology, University of Science and Technology of China. His current research interests include mobile computing, information security, and trusted computing and formal methods.

Zhongying Qiao received the bachelor's degree in electronics and electrical engineering from the University of Glasgow, U.K., and the joint master's degree in computer science from Technische Universität Berlin, Germany, and the University of Trento, Italy. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Notre Dame. Her research focuses on security, privacy, and big data analysis.



Jianwei Qian received the B.E. degree in computer science and technology from Shanghai Jiao Tong University, China, in 2015. He is currently pursuing the Ph.D. degree in computer science with the Illinois Institute of Technology, USA. His research interests include privacy and security issues in big data and social networking.

Linlin Chen received the B.E. degree from the Department of Computer Science and Technology, University of Science and Technology of China, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Illinois Institute of Technology, USA. His research interests span privacy and security issues in big data, machine learning, and mobile economics.



Junze Han received the B.E. degree from the Department of Computer Science, Nankai University, Tianjin, in 2011. He is currently pursuing the Ph.D. degree in computer science with the Illinois Institute of Technology. His research interests include data privacy, mobile computing, and wireless networking.

Jiahui Hou received the B.E. degree in computer science and technology from the University of Science and Technology of China, China, in 2015. She is currently pursuing the Ph.D. degree with the Department of Computer Science, Illinois Institute of Technology, USA. Specifically, she focuses on privacy-preserving machine learning. Her research interests include privacy and security issues in big data and mobile computing.