A Reversible Sketch Based on Chinese Remainder Theorem: Scheme and Performance Study

B.S. Adiga, Ravishankara Shastry, M. Girish Chandra and M.A. Rajan
Innovation Labs, Tata Consultancy Services, Bangalore, INDIA

Summary
In recent times, sketch based techniques are emerging as useful data stream computation techniques towards processing massive data. In many applications, finding heavy hitters and heavy changers is essential and this task demands reversibility property of sketches. Continuing the trend of arriving at newer reversible sketch, this paper presents a scheme based on Chinese Remainder Theorem. The scheme also involves the usage of two sketches based on two different prime-number sets for reducing the false positives. An attempt is made to present the technique on sound arguments by bringing in the relevant concepts from the existing literature. Simulation results, including how the data distribution affects the performance of the scheme, are presented as well.

Key words:
Data streams; sketch; heavy hitters; reversible sketch; Chinese remainder theorem; Zipfian data distribution

1. Introduction

Enormously large data sets are getting generated in our activities of recent times, like medical imaging, surveillance, network monitoring, etc. Coupled with the gathering is an ever-increasing demand for finer data analysis in scientific, engineering, and industrial applications [1]. One description of modern massive data sets is the *data stream*, which consists of data records or items. Data streams possess three challenging properties [1]: They represent (either explicitly or implicitly) huge amounts of data, they are distributed either spatially or temporally, and we receive or observe them at enormously high rates. For example, network service providers collect logs of network usage (telephone calls or IP flows) in great detail from switches and routers and aggregate them in data processing centers. They use this data for billing customers, detecting anomalies or fraud, and managing the network. In the stream data, each item of the form $(\text{key}, \text{value})$ arrives one after the other. Associated with each key is a time-varying *signal* or time series (Section 2 brings out these concepts more formally). The key is the identity of an item and the set of all keys of interest define the key space or domain size of stream (say, of size *N*). In the scenarios where data stream model is considered, the domain is discrete and is of very large value. For example, IP flows indexed by five-tuples [2]: source IP address (32 bits), source port (16 bits), destination IP address (32 bits), destination port (16 bits), and protocol (8 bits) can result in the domain size of 2^{100}. That is, for this example, a key belongs to the set $\{0,1,\cdots,2^{100}-1\}$.

In most cases, we cannot accumulate and store all of the detailed data [1]. We can archive past data but it is expensive to access these data. We would rather have an *approximate*, but reasonably accurate, representation of the data stream that can be stored in a small amount of space. It is not realistic to make several passes over the data in the streaming setting. It is crucial that we compute the summary representation or synopsis on the stream directly, in *one pass*. The emerging field of *data stream computation* deals with various aspects of computation that can be performed in a space- and time-efficient fashion when each item in a data stream can be touched only once (or a small number of times) [3]. In recent years a number of *synopsis structures* have been developed, which can be used in conjunction with a variety of mining and *query* processing techniques in data stream processing. Some of the examples of query include [4]: How many distinct IP addresses use a given link currently or anytime during the day? What are the top 5 voluminous flows currently in progress in a link? Are traffic patterns in two routers correlated? What are (un)usual trends?, etc. At any moment a synopsis can be used to (approximately) answer certain queries over the original data. Some key synopsis methods include those of sampling, wavelets, *sketches* and histograms [5].

Sketch based techniques are emerging as useful candidates in situations where it is required to handle many time series underlying the data stream (as in network monitoring application) towards "near real time" or online detection of appropriate events (by monitoring the whole data). Additionally, where hardware and energy constraints play a dominant role as in sensor networks, sketch based techniques would be definitely of great utility [5]. Sketches employ hashing techniques to approximate the "count" associated with an arbitrary key in a data stream using fixed memory resources. They are extraordinarily space efficient, and require space which is logarithmic in the number of distinct items in the stream domain. Some of the popular sketches developed and well

Manuscript received August 5, 2011
Manuscript revised August 20, 2011

studied in the literature include Count Min (CM) sketch [6] and CR-PRECIS [7].

Sketch has been recognized as a powerful technique for monitoring and analyzing of networks, as it facilitates real-time analysis of the massive, high speed data (traffic) generated by the networks. The two significant patterns which have many applications in accounting and anomaly detection in this context are (i) heavy hitters and (ii) heavy changers [2]. A heavy hitter is a key whose traffic volume (or any other quantity depending on the context, e.g. number of connections) exceeds a predefined threshold. Whereas a heavy changer is a key whose change in traffic volume between two monitoring intervals exceeds a pre-defined threshold. Needless to say, heavy-hitters need not necessarily correspond to flows experiencing significant changes. Further, in order to understand the impact of different heavy hitters and heavy changers, it is also useful to find the associated value (magnitude) of each of the heavy hitters and changers. Sketches mentioned earlier, the CM and CR PRECIS are *irreversible*, meaning that it is computationally infeasible to recover all heavy keys (heavy hitters or changers) using only the sketch-based summaries. These sketches do not contain information about what keys appeared in the stream. They can tell for a *given* (input) key, whether that input key is heavy with high accuracy. This irreversibility holds even if the non-heavy keys have negligible values [2]. In order to negotiate this problem, reversible sketch techniques, like, (i) reversible sketch using "modular hashing" and "IP mangling" [8] (ii) deltoids together with group testing [9], (iii) SeqHash [2] and (iv) XOR-based hashing [10] are proposed.

In this paper, we propose a novel reversible sketch based the Chinese Remainder Theorem (CRT). Rather than using the semi-ad hoc procedures of the existing reversible sketches, the proposed scheme is based on sound and elegant apparatus of number theory. In the direction of reducing false positives for the case of more than two heavy keys, two sketches based on two different sets of prime numbers is suggested. It is well known in the sketch-related literature that the performance of a sketch technique is strongly related to and influenced by the data distribution. It is interesting to study this dependency in the context of the proposed sketch, which adds to the novelty of the paper. The paper is organized as follows: In Section II, just enough background information is captured. The proposed CRT based reversible sketch is systematically discussed in Section III. Section IV touches upon few remarks on the dependency of performance of sketches on data distribution. Some simulation results and relevant discussion occupy Section V. Section VI concludes the paper.

2. Data Stream Model and CRT Based Sketch

2.1 Data Stream Model

As mentioned earlier, the streaming is a sequence of items of the form $(\text{key}, \text{value})$, arriving one after the other. Little more formally, a data stream *S* of running length *n* is a sequence of tuples:

$$S = (k_1, v_1), (k_2, v_2), \cdots, (k_n, v_n) \quad (1)$$

The above model is very general and one can instantiate it in many ways with specific definitions of the key and updates. For example, as mentioned in Section I, the key can be defined using one or more fields in packet headers such as source and destination IP addresses, source and destination port numbers, protocol number etc. The update can be the size of a packet, the total bytes or packets in a flow etc. Associated with each key k_i is a time-varying *signal* $U(i)$. The arrival of each new data item (k_i, v_i) causes the underlying signal to be updated by $v_i : U(i) += v_i$. The model discussed so far is the most general model referred to as the Turnstile Model. There are other special cases of this model as well. When the value v_i is a positive number, which results in monotonically increasing key counts, we end up with what is called as Cash Register Model. In the time-series model the stream *S* defines the signal directly, i.e. *i*th update changes $U(i)$.

2.2 The Chinese Remainder Theorem (CRT)

The CRT is essentially an analytical process of calculating dividend from remainders [11]. Assume that unknown integer dividend is *x*. Let dividing it by pair-wise co-prime numbers $p_1, p_2, \cdots p_d$ (the pair-wise co prime is simply $\gcd(p_i, p_j) = 1$ for $i \neq j$) resulted in the remainders $m_1, m_2, \cdots m_d$, that is, we have simultaneous congruences given by :

$$x \equiv m_i \pmod{p_i} \text{ for } i = 1,2,\cdots,d \quad (2)$$

The CRT suggests that if $x < N = \prod_{i=1}^{d} p_i$, it can be uniquely determined with the following:

$$x \equiv \left(\sum_{i=1}^{d} \delta_i m_i \right) \pmod{N} \quad (3)$$

where δ_i is obtained as follows: First, obtain $Q_i = \frac{N}{p_i}$ and then obtain the modular inverse q_i of Q_i, i.e. $Q_i q_i \equiv 1 (\mathrm{mod}\ p_i)$. Finally, arrive at δ_i as $\delta_i = Q_i q_i$. Modular inverse can be calculated by extended Euclidean algorithm or other improved algorithms developed over the years.

2.3 CR PRECIS Sketch

The CR-PRECIS sketch is extensively studied by Ganguly and Majumder in [7]. The CR-PRECIS data structure is depicted in Fig.1 and this two-dimensional array *C* is made of row of counters or "buckets" for each of the *d* hash functions. Each row is sometimes referred to as the hash table and hence we have *d* hash tables. The hash functions are rather simple and take the form

$$h_i(x) = x \bmod p_i \text{ for } i = 1, 2, \cdots, d \qquad (4)$$

The number of buckets in each hash table is chosen from the consecutive primes $p_1, p_2, \cdots p_d$; hence we have unequal number of buckets in each row (or table). It is useful to note that compared to the randomness involved in the hash function generation in other sketches like CM, the CR-PRECIS has a *deterministic* flavor.

The update involved in the sketch is straightforward. When a data stream item (k_t, v_t) arrives, the incoming key k_t is hashed to a bucket in each row based on the remainders obtained by dividing k_t by $p_1, p_2, \cdots p_d$:

for $i = 1$ to d
$$C[i][k_t \bmod p_i] += v_t$$
end

Essentially, in the CR-PRECIS, we are using the CRT to represent $k_t \bmod N$ by $m_{t1}, m_{t2}, \cdots m_{td}$ where $k_t \equiv m_{ti} \bmod p_i$. As mentioned earlier, with the arrival of an item (k_t, v_t) the buckets represented by $m_{t1}, m_{t2}, \cdots m_{td}$ are chosen for update by obtaining the remainders. Similarly, for the point query about k_t at a particular instant of time, we use the so far accumulated values in the counters pointed by $m_{t1}, m_{t2}, \cdots m_{td}$. These values are "aggregated" [12], for example, by evaluating median, minimum, etc to arrive at the answer for the query. This aggregation or any general estimation is essential for arriving at an accurate result due to the problem of "collision". The latter refers to the fact that multiple keys may hash to the same bucket, due to the many-to-one mapping from $\{0, \cdots, N-1\} \to \{0, \cdots, p_i - 1\}$. In other words, in each of the relevant buckets for the key k_t, there is contamination by other keys. Thus, when the incoming update values are non-negative, the hash table counts will over-estimate the true count, whereas when the incoming updates are either positive or negative (deletions), the hash table count could be either an over-estimation or an under-estimation [5]. In either case, the use of the median among the counts provided by the different hash functions for the given item provides an estimate in terms of well proven theoretical guarantees [5]. However, when the stream follows cash-register model with positive "counts" v_i, minimum provides better estimate.

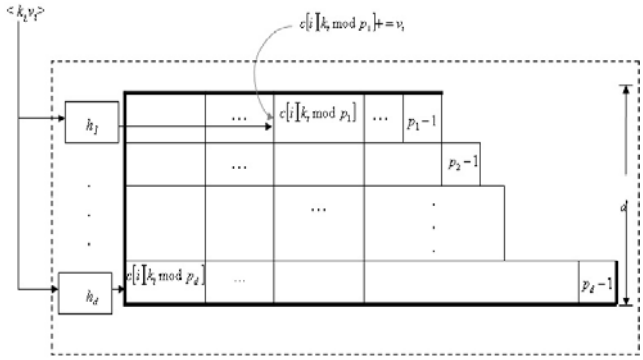

Fig.1 CRT Based Sketch Data Structure

3. CRT Based Reverse Sketching

As brought out in Section 1, sketches exhibiting reversible property are of great utility in certain applications. Essentially, here we are interested in identifying the heavy keys: heavy hitters or heavy changers. In the rest of the paper, we will be restricting the discussion to heavy hitters. In the direction of identifying heavy hitters, an intermittent notion of *heavy buckets* is generally used. A bucket is called heavy if its counter value (i.e., sum of values of all keys hashed to the bucket) crosses a predetermined threshold. It is easy to see that *for any heavy hitter, every bucket that it falls into, in each of the d tables, is a heavy bucket*. A candidate set Ω of heavy hitters refers to the set of keys whose buckets within the *d* hash tables are all heavy buckets. It is to be noted that Ω is the *superset* of the actual heavy hitters, and it may contain some *non*-heavy hitters that happen to fall into heavy buckets in all *d* hash tables (usual collision).

As mentioned in the previous section, in creating the CR-PRECIS sketch we take the direction of going from k_t to $m_{t1}, m_{t2}, \cdots m_{td}$. This direction of going from k_t to $m_1, m_2, \cdots m_d$ is easy. For reverse sketching, we have to move from $m_{t1}, m_{t2}, \cdots m_{td}$ to k_t. The mathematical apparatus of CRT provides an elegant way to traverse in this direction through Eqn.3. Thus, it is quite logical to think of reverse sketch based on CRT. In the direction of presenting the proposed reverse sketch we consider two cases: one heavy hitter and multiple heavy hitters.

3.1 One Heavy Hitter Case

The case of one heavy hitter is rather trivial. In this case each hash table has got one heavy bucket. Effectively, we identify the heaviest bucket (highest-valued counter) in each of the hash tables. Thus obtained counter "addresses" or numbers $m_{h1}, m_{h2}, \cdots m_{hd}$ are then utilized in Eqn.3 to arrive at the heavy hitter k_{ht}.

3.2 Multiple Heavy Hitters Case

In the case of two or more heavy hitters, we end up in having more than one heavy bucket in each of the hash tables. The heavy buckets might have been identified by comparing the count values to a threshold. We propose to use *two* sketches based on different sets of prime numbers for this scenario. *The idea is to obtain the set of "suspect" heavy hitter keys* $\Omega 1$ *and* $\Omega 2$ *from each of the sketches and then take the intersection of these two sets.* Each of the sets $\Omega 1$ and $\Omega 2$ contain many false positive keys, i.e. keys which are detected as heavy, but actually are not. One proven way to reduce false positives in reverse sketches is to have more buckets and hence more memory [2]. In our scheme, the false positive keys which result in individual sketches are going to get dropped in the process of intersection ($\Omega 1 \cap \Omega 2$). Effectively, we have a systematic way of reducing/eliminating the false positives at the cost of additional memory (required for the second sketch). It is also worth noting that since each sketch shuffles heavy hitters across different sketch entries, approximate agreement among sketches can be used to robustly detect the heavy hitters. An example can clarify the false positives and the overall scheme itself in a better way.

A data stream of size 1000 elements generated using Zipfian distribution (see the next section) with the parameter value of 1.3, is CR sketched with the following two sets of prime numbers:

$$P1 = \{37, 43, 47, 53, 59, 61\}$$

and

$$P2 = \{23, 29, 31, 37, 41, 43\}$$

Considering the case of top two heavy hitters, the two heavy buckets in each of the hash tables and each of the sketches are identified. They are as follows:

$$\overline{H}_{11} = \{2, 29, 20, 48, 41, 21\}$$
$$\overline{H}_{12} = \{4, 15, 40, 43, 23, 42\}$$
$$\overline{H}_{21} = \{10, 22, 11, 2, 16, 29\}$$
$$\overline{H}_{22} = \{20, 15, 22, 4, 32, 15\}$$

where $\overline{H}_{21} = \{10, 22, 11, 2, 16, 29\}$ is the set of first highest heavy buckets in CR sketch 2, etc. Now, similar to the existing methods in the literature, we have to take different possible combinations of the heavy buckets in the respective sketches. This leads to $2^6 = 64$ different sets of heavy buckets for each sketch. Applying CRT in the form Eqn.3, the following two sets of "suspect" keys can be generated:

$\Omega 1 =$ {631.0, 5611864927.0, 11604194599.0, 2952570476.0, 11841387243.0, 3189763120.0, 9182092792.0, 530468669.0, 8497397987.0, 14109262283.0, 5838103536.0, 11449967832.0, 6075296180.0, 11687160476.0, 3416001729.0, 9027866025.0, 995127730.0, 6606992026.0, 12599321698.0, 3947697575.0, 12836514342.0, 4184890219.0, 10177219891.0, 1525595768.0, 9492525086.0, 840900963.0, 6833230635.0, 12445094931.0, 7070423279.0, 12682287575.0, 4411128828.0, 10022993124.0, 4240497188.0, 9852361484.0, 1581202737.0, 7193067033.0, 1818395381.0, 7430259677.0, 13422589349.0, 4770965226.0, 12737894544.0, 4086270421.0, 10078600093.0, 1426975970.0, 10315792737.0, 1664168614.0, 7656498286.0, 13268362582.0, 5235624287.0, 10847488583.0, 2576329836.0, 8188194132.0, 2813522480.0, 8425386776.0, 154228029.0, 5766092325.0, 13733021643.0, 5081397520.0, 11073727192.0, 2422103069.0, 11310919836.0, 2659295713.0, 8651625385.0, 1262.0}

and

$\Omega 2 =$ {631.0, 1317415009.0, 98691952.0, 67324943.0, 692618100.0, 661251091.0, 791309421.0, 759942412.0, 522109555.0, 490742546.0, 620800876.0, 589433867.0, 1214727024.0, 1183360015.0, 1313418345.0, 1282051336.0, 418587958.0, 387220949.0, 517279279.0, 485912270.0, 1111205427.0, 1079838418.0, 1209896748.0, 1178529739.0, 940696882.0, 909329873.0, 1039388203.0, 1008021194.0, 284532964.0, 253165955.0, 383224285.0, 351857276.0, 996926004.0, 965558995.0, 1095617325.0, 1064250316.0, 340762086.0, 309395077.0, 439453407.0, 408086398.0, 170253541.0, 138886532.0, 268944862.0, 237577853.0, 862871010.0, 831504001.0, 961562331.0, 930195322.0, 66731944.0, 35364935.0, 165423265.0, 134056256.0, 759349413.0, 727982404.0, 858040734.0, 826673725.0, 588840868.0, 557473859.0, 687532189.0, 656165180.0, 1281458337.0, 1250091328.0, 31368271.0, 1262.0}

Due to the combinations of elements of heavy bucket sets, one can see large number of false positives in each of the suspect key sets. They can be reduced by taking the intersection of $\Omega 1$ and $\Omega 2$:

$$\Omega 1 \cap \Omega 2 = \{631.0, 1262.0\}.$$

As a cross-check, the results are compared with actual frequency of heavy hitter elements:

Sl.No.	ID	Sketch1	Sketch2	Actual	Error1	Error2
0	631.0	261	264	258	3	6
1	1262.0	100	102	94	6	8

The above table shows the estimate of the counts based on using the minimum aggregation rule for each of the sketches. The actual values for the two heavy hitters are also shown; the accuracy in this toy example is very high as for just 1000 keys we have used 300 and 204 counters in sketch 1 and sketch 2 respectively.
Similar observations and arguments can be built for more than two heavy hitters.

4. Effect of Data Distribution on the Performance of the Sketch

As noted, the CR-PRECIS sketch (and CM Sketch as well) uses hash functions which map the signal coordinates $U(i)$ to rows of the measurements. Further, as noted earlier, the inherent problem with sketches is that keys may collide, namely, hash to the same bucket, producing errors in the estimated counts. What if the signal has only *s* large (or non-zero) entries, i.e., exhibit *s*-sparsity? Elaborating slightly, $U = [U(0) \;\; U(1) \;\; \cdots \;\; U(N-1)]$ has *s* large values. It is well known that if hash function *h* is chosen uniformly at random from a pre-specified family of hash functions, then the probability that positions i and i' are hashed into the same bucket is low [12]. Using similar arguments, it is possible to show that for *s*-sparse signals which are hashed into more than *s* buckets, then with a high probability, a large fraction of the significant entries are hashed into separate "measurements". The term "measurement" has been brought into context to spell out the fact that sketching is closely linked to Compressive Sensing (CS); see [12] and [13] for more details. In fact, implicit in many of the guarantee proofs is the ability of the hash functions to isolate a few significant signal values from one another [12]. Since the sparsity can be linked to the data distribution, it is customary to find sentences like "the performance of CM sketch is strongly related to and influenced by the skew of the data distribution [6]; when the skew of the data distribution is larger, its efficiency is better, and vice versa". In many applications, the signal entries follow Zipfian, or power law, distributions; here the (relative) frequency of the *i*th most frequent item is proportional to i^{-z}, for some parameter *z*, where *z* is typically in the range 1 to 3 ($z = 0$ gives a perfectly uniform distribution). In such cases, it makes sense to use the skew in the distribution to show a stronger space/accuracy tradeoff [6].
The arguments captured so far revolve around random hash functions. But, in CRT based sketching we use deterministic hash functions. With modular operations in place for each of the hash functions (see Eqn.4), there is an inherent random behavior associated with the hash functions. At this juncture, it is worth noting the following statement from [14]: Choosing *x* randomly in $\{0, \cdots, N-1\}$ is equivalent to, independently choosing m_i randomly from $\{0, \cdots, p_i - 1\}$ for different *i* and then determining *x* from the system of simultaneous congruences. Thus, one can expect CRT based sketches to behave similarly as the random hash function based sketches when it comes to data distribution. The results of next section support this aspect of CRT based reverse sketching

5. Simulation Results and Some Remarks

Elaborate simulations were carried out towards studying the performance of the proposed sketch. The experimentation is carried out with different distributions of data (including different Zipfian parameter *z*), different sets of prime numbers (hence different number of total buckets) and different number of heavy hitters.
Fig.2, Fig.3, Fig.4 and Fig.5 depict some typical error performance curves for the case of two and four heavy hitters respectively. For each heavy hitter, the curves corresponding to both randomization and no randomization of keys are provided (note that randomization corresponds to "mangling" [8]). In all these figures, the squared error is averaged over 50 different realizations of data. As expected, the error performance improves when the data exhibits "better sparseness", corresponding to larger values of *z* in the closed interval [1, 3].

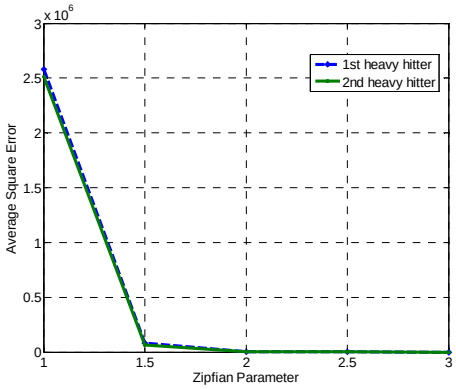

Fig.2. Error Performance for Two Heavy Hitters Case; No Randomization of Keys

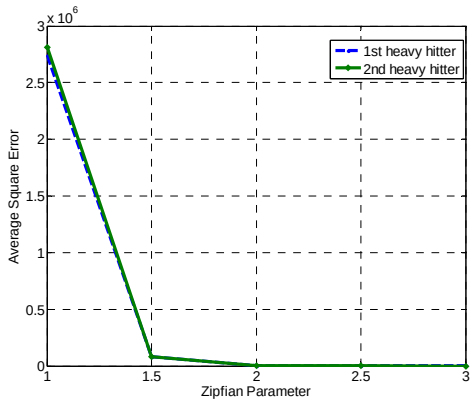

Fig.3. Error Performance for Four Heavy Hitters Case; Keys are Randomized

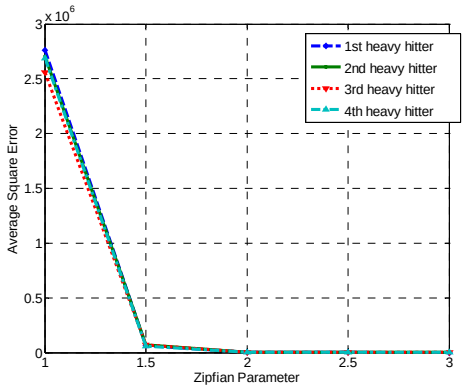

Fig.4. Error Performance for Four Heavy Hitters Case; No Randomization of Keys

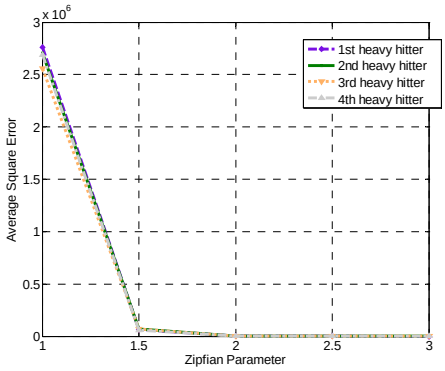

Fig.5. Error Performance for Four Heavy Hitters Case; Keys are Randomized

The following remarks are worth making about the proposed novel CR based reverse sketch:

(1) The reversing method is very elegant and systematic.

(2) As a further extension, it is possible to arrive at clever ways of combining heavy buckets, reducing both complexity and number of false positive candidates.

(3) It is interesting to explore the use of regression formulation suggested in [2] with the proposed reverse sketch for estimating the values (magnitudes) of the heavy keys. Estimation of values of heavy keys is important for two reasons. First, when the number of heavy keys is large, it is desirable to highlight the most important heavy keys with the highest values. Second, using the estimated values, one can further reduce the false positive rate by eliminating those non-heavy keys included in the candidate set of heavy keys. Through experimental studies, the authors have shown that by using estimation, it is possible to reduce the false positive rate significantly at the expense of only a small increase in the false negative rate. Also, an account of the noise values due to non-heavy keys that are determined by the underlying traffic behavior is taken during estimation.

(4) Since keys are directly hashed or since no modular hashing [8] is adopted, mangling is not required. That is, direct hashing some how gets the "correlation" among the adjacent keys compensated in the modular operations. Simulation results did confirm this (Fig.2, Fig.3, Fig.4 and Fig.5).

6. Conclusions

In this paper, we systematically presented an elegant reversible sketch based on the Chinese Remainder Theorem. Useful simulation results in terms of bringing out the dependency of the performance of the proposed sketch on data distribution are presented as well. Efforts are under way towards applying the technique in some suitable applications.

References

[1] A.C. Gilbert and M.J. Strauss, "Group Testing in Statistical Signal Recovery", Technometrics, Vol. 49, No. 3, August, 2007, pp. 346-356

[2] T. Bu, J. Cao, A. Chen and P. Lee, "Sequential hashing: A flexible approach for unveiling significant patterns in high speed networks", Elsevier, Computer Networks, No.54, July 2010, pp.3309-3326

[3] B. Krishnamurthy, S. Sen, Y. Zhang and Y. Chen, "Sketch-based Change Detection: Methods, Evaluation, and Applications", Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement October 2003, USA, pp. 234-247

[4] Luca Becchetti, "Compact data structures: data streaming", June 2011, www.dis.uniroma1.it/~becchett/Elective/slide/stream.pdf

[5] C.C. Aggarwal and P S. Yu, "A Survey on Synopsis Construction in Data Streams", Book Chapter in Data Streams: Models and Algorithms, Ed. C.C. Aggarwal, Springer, Jan.2007

[6] G. Cormode, "Count-Min Sketch", Encyclopedia of Database Systems, Springer, 2009, pp. 511-516.
dimacs.rutgers.edu/~graham/pubs/papers/cmencyc.pdf

[7] S. Ganguly and A. Majumder, "CR-PRECIS: A deterministic summary structure for update data streams", in ESCAPE, 2007, pp.48-59

[8] R. Schweller, A. Gupta, E. Parsons and Y. Chen, "Reversible Sketches for Efficient and Accurate Change Detection over Network Data Streams", Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, Italy, Oct. 2004

[9] G. Cormode and S. Muthukrishnan, "What's New: Finding Significant Differences in Network Data Streams", Proc. of IEEE INFOCOM, Hong Kong, March 2004, pp. 1534 -1545.

[10] W. Feng, Q. Guo, Z. Zhang and Z. Jia, "Reversible Sketch Based on the XOR-based Hashing", IEEE Asia-Pacific Conference on Services Computing (APSCC 06), Guangdong, Dec.2006, pp.93-98.

[11] C. Wang, Q. Yin and W. Wang, "An efficient ranging method based on Chinese remainder theorem for RIPS measurement", Science China-Information Sciences, Vol.53, No.6, June 2010, pp. 1233-1241

[12] A. Gilbert and P. Indyk, "Sparse Recovery Using Sparse Matrices", Proc. of IEEE, Vol.98, Issue 6, June 2010, pp. 937-947.

[13] B.S. Adiga, M. Girish Chandra, Ravishankara Shastry and Swanand Kadhe, "Data Streams and Sketching", TCS Technical Report, July 2011.

[14] Johan Hastad, Lecture Notes, "Advanced Algorithms", Jan.2000

B. S. Adiga obtained his BE (Electrical Engg.) and MTech (Industrial Electronics) degrees from Karnataka Regional Engineering College, Surathkal, India. He obtained his PhD in Computer Science from the Indian Institute of Science, Bangalore, India. He worked as a scientist at National Aerospace Laboratories, Bangalore, India, for nearly 20 years. Later on, he was with Motorola India Electronics Limited and Philips Innovation Labs, Bangalore, India. Presently, he is a Principal Scientist at the Innovation Labs, Tata Consultancy Services, Bangalore, India. His interests are in the broad areas of Signal Processing, Communications and Computing including, Error Control Coding, Compressive Sensing, High-Performance Computing and Multimedia Signal Processing.

Ravishankara Shastry obtained his Masters degree from VTU Belagaum in Computer Applications in 2002 and Masters degree from Chennai Mathematical Institute (CMI) Chennai in Theoretical Computer Science in 2010. Prior to join TCS Innovation Labs, he worked as Software Engineer in IBM India Software Labs(IBM ISL) Bangalore. Currently, he is a researcher at the Innovation Labs, Tata Consultancy Services, Bangalore, India. His interests are in algorithms for massive data sets, graph theory, combinatorics, Machine learning and data mining in data streams.

M. Girish Chandra obtained his BE in Electronics from University Visvesvaraya College of Engineering, Bangalore and MTech from IIT Madras in Communication Systems and High Frequency Technology. He earned his PhD as a Commonwealth Scholar in Digital Communication from Imperial College, London. Presently, he is a Senior Scientist at the Innovation Labs, Tata Consultancy Services, Bangalore, India. Earlier, he was holding the position of Assistant Director at the Aerospace Electronics Division of National Aerospace Laboratories, Bangalore, India. His interests are in the broad areas of Communications and Signal Processing, including, Error Control Coding, Compressive Sensing, Cross-Layer Design and Multimedia Signal Processing.

Rajan M.A has B.E., M.Tech. and PhD Degrees in Computer Science and Engineering, and M.Sc., M.Phil and PhD in Mathematics. During 2000-2005, he worked at the ISRO Satellite Centre (ISAC), Bangalore, INDIA as a Scientist and was actively involved in realization of several spacecrafts. From September 2005 onwards, he is with Innovation Labs, Tata Consultancy Services, Bangalore as Research Scientist. His areas of interest include Computer Networks, Cross Layer Design, Number Theory, Graph Theory, Combinatorics, Coding Theory and Functional Analysis. He has published several research papers in national and international conferences and journals.