parallelism to increase the efficiency of FADS algorithm and make it applicable for big data stream anonymization. Second, proposing of a simple proactive heuristic estimated-round-time to prevent publishing of a tuple after its expiration. Third, demonstrating (through experimental results) that FAST is more efficient and effective over FADS and other existing algorithm while it noticeably decreases the information loss and cost metric during anonymization process.

The rest of the paper is organized as follows: Section 2 reviews the related work. Section 3 formally defines the anonymization model and information metrics. FAST algorithm is described in Section 4. Empirical simulation is explained in Section 5. Finally, Section 6 concludes the paper and discusses the future works.

2. RELATED WORK

Most of the proposed approaches to preserve the privacy of data are focused on the privacy of relational and static data. Privacy models such as k-Anonymity [13], l-Diversity [11], ϵ-Differential Privacy [4], t-closeness [10] and so on, are proposed to cope with attribute linkage, record linkage, table linkage and probabilistic attack [5]. As data streams are potentially infinite, fast flowing, and rapidly changing the privacy-preserving model for static data are not applicable for data for data stream [1]. The data stream anonymization approaches fall into four main categories: perturbation-based, tree-based structure, counterfeit value-based, and cluster-based.

Li et al. in [9] presented a method under the additive random perturbation framework, which maximally preserve the privacy of data streams given a fixed utility. However, too much artificial noise makes the anonymized data more difficult to analyze. Another weakness of the algorithms is that it can only handle numerical data.

Apply of tree structures for data anonymization is proposed by Zhou et al. in [15]. A single tree contains multiple nodes that represented the tuples in the stream. The tree structure is changed when new tuples arrived or the number of the tuples reach to k. When the size of a tree is k, one tuple should be released. These approach time and space complexity is $O(|S|\delta \log \delta)$ and $O(\delta)$ respectively [7]. The information loss rate of this approach is also high.

Kim et al. in [8] proposed a delay-free anonymization framework, to protect data against potential privacy breaches. Unlike existing work, l-diverse artificial sensitive data are generated and added to main sensitive data instead of quasi-identifier attribute be generalized. Then late validation method will be used in order to reduce the generation of counterfeits. The main drawback of this approach is that adding counterfeit data lead to high overhead that is not tolerable in big data stream environments.

In the cluster-based approaches, each tuple is inserted into a cluster in a manner that each one contains at least k tuples. Then tuples will be published with cluster generalization according to their time constraint. The clusters are reused during data stream anonymization.

Cao et al.[3] proposed a method in which tuples are published continuously with their cluster's generalization before their time constraint is overstepped. Such methods consider two sets of clusters, i.e, k-Anonymization, which is reused for data publishing and none k-Anonymization, which is used to merge and split none k-Anonymized clusters to create new k-Anonymized clusters. By techniques the algorithm pursues, it accomplishes a low information loss rate compared to the other anonymization methods. Besides, the algorithm adheres the l-Diversity method. Unfavorably, the algorithm considers no cluster size limitations, which lead to growth of the cluster size linearly dependent to data stream size. Also the total complexity time is $O(S^2)$ that is too high to be efficient for a data streaming algorithm [7].

Zakerzadeh et al. [14] introduced a new cluster-based algorithm for anonymizing numerical data streams using window processing called FAANST. It takes three parameters: k, μ, and δ. k is the degree of anonymity. μ specifies the size of the processing window. And δ determines the accepted clusters which are kept to be reused in the later rounds. When the numbers of tuples in the processing window reaches μ, one round of the clustering algorithm is started. The window can slide again in order to accumulate more tuples in each round. The main drawback of FANNST is that some tuples may remain in the system more than allowable time constraint. In addition, the time and space complexity of the algorithm is $O(S^2)$ and not efficient for a data streaming algorithm [7]. Another weakness of FANNST is that it does not support categorical data.

Guo et al. in [7] presented an algorithm, FADS, for data stream anonymization, in which the time complexity of the approaches is $O(|S|)$, which is linear to the stream size also the space complexity is O(C), which is constrained by a constant C. the algorithm considers a set as a buffer and saves at most δ tuples in it. Also, another set (set_{kc}) is considered to hold the newly created cluster for later reuse. Each k-Anonymized cluster will be remained in set_{kc} up to the reuse constraint T_{kc} and after that the cluster is removed. The main drawback of the FADS is that the algorithm does not check the remaining time of tuples that hold in the buffer in each round and are outputted them when they might be considered to have expired. The other important weakness of FADS is that it is not parallel and cannot handle a large amount of data streams in tolerable time.

3. ANONYMIZATION MODELS

The k-anonymization model which is used for data stream is different with traditional k-anonymization models in some aspects. In this section, a k-anonymization data stream model is defined formally.

3.1 Anonymization Model

Definition1. Data stream

A sequence of tuples is defined as $< s_n >_{n \in N}$ where N is the natural number set. The kth term of $< s_n >$ is the ordered pair (k, t_k) where k is a number and tk is a tuple. The length of a finite sequence is the number of terms it contains. An infinite sequence is a sequence whose length is infinite. A data stream S is a potentially infinite sequence of tuples, depicted by $< t_i >$, where all tuples ti follow the schema $t_i =< ID, a_1, , a_m, q_1, , q_n, TS >$. ID is an identifier attribute; $q_1, , q_n$ are quasi-identifiers, a_1, a_m are other attributes, and TS is a time stamp.

Definition 2. k-anonymized data stream function
Suppose that S and Sáre data streams. Let P_s denotes the set of terms in S. The function $A : P_s \rightarrow P'_s$ is a k-anonymized data stream function if the following property is hold:

$$\forall t \in S, \exists t' \in S' where A(t) = t'$$
$$\forall t' \in S', |EQ(t')| \geq K \quad (1)$$

$EQ(t')$ is an equivalent class function with respect to function A and defined over stream S as follow:

$$EQ(t') = \{t \in S \mid A(t).q_i = t'.q_i, i = 1, \cdots, n\} \quad (2)$$

Sís called a data stream satisfying k-anonymity [7]. Our goal is to define a k-anonymized data stream function which generate S' as soon as possible. We called this function as FAST. To this goal, similar tuples are partitioned into a cluster. Then the tuples in a cluster published with same generalizations, which is called the clusterś generalizations.

Definition 3. Cluster
Cluster is a set of tuples in a stream. Suppose that P_S is a set of tuples in stream S. Cluster C can be defined as follow:

$$C = \{t \mid t \in P_s\} \quad (3)$$

Definition 4. Cluster generalization
Generalization is a function that maps a cluster into a tuple. More formally, generalization function G is defined as $G : PowerSet(TUPLE) \rightarrow TUPLE$ where $TUPLE$ is the set of all possible tuples. Note that $PowerSet(TUPLE)$ is the set of all possible clusters. Now G can be described as follows:

$$G(c) = gtwhere(cisacluster)and(gtisatuple)and$$
$$\forall t \in c, \forall q \in QID, t.q \sqsubseteq gt.q \quad (4)$$

QID is the set of quasi-identifiers. Suppose that q_1 and q_2 are the same tuple attributes, then $q_1 \sqsubseteq q_2$ iff

$$\begin{cases} q_1 \subseteq q_2 & if\ q_1\ and\ q_2\ are\ numerical \\ q_2 \in Anssestor(q_1) & if\ q_1\ and\ q_2\ are\ categorical \end{cases} \quad (5)$$

As an instance, consider the cluster C=<"prof.young", Academic ,43> ,<"Mr.Zhou",non-Academic,39> , "Prof.Chung" ,Academic,46 >. This cluster can be generalized to $gc =< *, staff, [39-46] >$.

There are two common techniques to anonymizing data: generalization and suppression.

Generalization is based on attribute values. There are two types of attributes; numerical and categorical. In order to generalize numerical attribute of a tuple, the attribute value is mapped to an interval that covers the value. For example, consider $t =<$ "$Prof.Young$", $academic, 43 >$ where 43 presents the age of "Prof.Young" who is the academic staff. The age, 43, is a numerical attribute and can be generalized into the interval $[41-44]$. A categorical attribute, is mapped to a node in a domain generalization hierarchy (DGH). For generalization, each node can be replaced with its parent according to DGH. For example, based on DGH

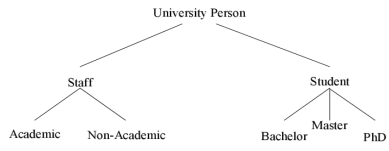

Figure 2: University-Person DGH

depicted in Figure 2 academic can be generalized to staff and University-Person.

Suppression can be considered as most degrees of generalization. For example in Figure 2, academic is suppressed to University-Person.

Definition 5. k-anonymized cluster If a cluster C built from data stream and the number of unique tuple in the cluster is greater than k, the cluster is called a k-anonymized cluster.

Definition 6. Information loss Generalize a cluster to a tuple may cause information lost, because generalization function is invertible. Suppose that $t.QID$ is a vector $v =< q_1, , q_n >$ where q_i is a quasi-identifier attribute. The information loss rate can be calculated as follow:

$$infoloss(c, G) = w \times infoloss(G(c).QID) \quad (6)$$

where c is a cluster, G is a generalization function, and $w = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$ is a weighted vector of size $n * 1$. w_i is the weight of ith quasi-identifier attribute, and $\sum_1^n w_i = 1$. The vector w shows the importance of each quasi-identifier. Consider t.QID, then:

$$infoloss(t.QID) = [i_1, \cdots, i_n] where$$
$$l_i = \begin{cases} \frac{|q_i|}{|domain(q_i)|} & where\ q_i\ is\ numerical \\ \frac{|Leaves(N_i-1)|}{|Leaves(DGH_i-1)|} & where\ q_i\ is\ categorical \end{cases} \quad (7)$$

$|.|$ is the size function. Domain of qi attribute is represented by $domain(q_i)$. N_i is the node corresponding to q_i in the related DGHi. $Leaves(N_i)$ is the number of leaves of the sub tree rooted at N_i. $Leaves(DGH_i)$ is the number of leaves of DGH_i. To explain the intuition behind $infoLoss(c, t)$ suppose that $G(c)$ is published instead of each tuple in c. $infoLoss(c, t)$ rate of information will be lost because it is impossible to generate original tuples from the published ones. The goal is to choose c in such a way when $G(c)$ is published instead of c'tuple, the information loss be minimized.

Definition 7. Cost
In big stream data publications, it is very important to publish data with low latency. The function $Cost()$ is defined to show the impact of latency in the publication of a data. This function is chosen as a goal function in our approach. α is the parameter that scale the effect of latency. $arrivalTime$ and $publishTime$ are the times when a data arrived and

published respectively.

$$Cost(c,G) = infoLoss(c,G) \times (1+a)^{(PublishedTime-arrivalTime)} \quad (8)$$

Definition 8. distance between two tuple
Suppose that we want to choose a cluster c of size k from a set of tuple Sett where $|Set_t| \geq k$ in such a way that information loss be minimized. According to [7], if the set of tuples are closest to each other, the information lost will be minimized. The distance between two tuples t_1 and t_2 is calculated by the following formula.

$$distance(t_1,t_2) = w \times distance(t_1.QID, t_2.QID) \quad (9)$$

where w is a $n*1$ weighted vector, and $t_1.QID$ is the vector of t_1's quasi-identifier. $distance(t_l.QID, t_k.QID)$ is defined as follow:

$$distance(t_l.QID, t_k.QID) = [d_1, , d_n] where$$

$$d_i = \begin{cases} \frac{|t_i.q_i \bigcap t_k.q_i|}{|domain(q_i)|} & where\ q_i\ is\ numerical \\ \frac{|Leaves(N_{lk})|-1}{|Leaves(DGH_i)|-1} & where\ q_i\ is\ categorical \end{cases} \quad (10)$$

$t_l.q_i \cap t_k.q_i$ is the subsection of $t_l.q_i$ interval and $t_k.q_i$. N_{lk} is the lowest common ancestor of $t_l.q_i$ and $t_k.q_i$.

4. FAST

4.1 Details of Algorithm

The details of FAST are given in Algorithm 1. The algorithm reads δ tuples continuously and passed them to new threads until the number of threads reaches to $MaxNumThread$. To publish data, each thread calls procedure $publish(Set_{tp})$. Then the first tuple in Set_{tp} is removed, named t, and procedure $PublishData(Set_{tp}, t)$ is called. This procedure is shown in 2.

Procedure $PublishData(Set_{tp}, t)$ is main part of algorithm and tries to anonimyzed tuple t. At first, the procedure finds $t's$ k-1 nearest tuples in Set_{tp} and insert them in a new cluster is called C_{new} and generalize it into g_{cnew}. Then a reusable cluster with minimum information loss (C_{k-best}) that covers tuple t, is chosen from Set_kc. If C_{k-best} exist and has smaller information loss compared to C_new , tuple t is published with C_{k-best} generalization and time of C_{k-best} is updated. Then other k-1 tuples that remain in Set_{tp} are checked for whether they can process in another round or must be suppressed and published immediately. if tuple t does not match with any cluster in Set_{kc} which has less information loss than C_{new} , tuple t and it's neighbors are published with C_{new}' generalization (g_{cnew}). Then, g_{cnew} is inserted in Set_{kc}. The other tuples in Set_{tp} are checked for remaining time. If they have enough time to process, they are passed to Set_{tp} otherwise they will be suppressed and published.

In the following, a simple example is illustrated for better understanding. Assume that Table 1 is a portion of a university data stream, in which quasi-identifier are age and job. Also δ and k are assumed as $\delta = 3$ and$k = 2$. Suppose that in thread n the value of variables are as follows:

- $Set_{tp} = \{(<id_n, 45, academic>, <id_{n+1}, 26, Non-academic>, <id_{n+2}, 39, PhD>)\}$

Algorithm 1 FAST (S,k,δ,T_{kc},T_e,$MaxNumThread$)

while $S \neq \emptyset$ **do**
  Read δ tuples and insert them into Set_{tp};
  Remove the clusters that their existence time exceeds T_{kc};
  if (number of running threads in the system < $MaxNumThread$) **then**
    Create a new thread, pass Set_{tp} to it, and call function publish(Set_{tp}) from the thread;
  else
    Wait while a thread terminates;
  end if
end while

Algorithm 2 Publish(Set_{tp});

Remove the first tuple of Set_{tp} and call it t;
PublishData(Set_{tp});
Terminate the thread

Algorithm 3 PublishData(Set_{tp},t)

Select $k-1$ unique tuples which are the closest to t among tuples in Set_{tp} and insert them into Cluster C_{new}. Generalize C_{new} into g_{cnew}.
For each cluster C_{kc} which covers t, calculate information loss, and select a cluster incures less information loss. Call the cluster as C_{k-best}.
if C_{k-best} exists and C_{k-best} generate less information loss than g_{cnew} **then**
  Publish t with C_{k-best}'s generalization;
  Update round-time estimation;
  Synchronized (C_{k-best}) {Update C_{k-best} publish time;}
  for Each tuple tp in Set_{tp} **do**
    if (current-time - arrival-time + estimated round-time) < T_e **then**
      Synchronized (S) {Insert tp as the first element of S;}
    else
      Suppress and publish tp;
    end if
  end for
else
  publish C_{new} with g_{cnew};
  Update round-time estimation;
  Synchronized (Set_{kc}) {Insert g_{cnew} into Set_{kc} and set its publish time;}
  for Each tuple tp in (Set_{tp} - Set_{new}) **do**
    if (current-time - arrival-time + estimated round-time) < T_e **then**
      Synchronized (S) {Insert tp as the first element of S;}
    else
      Suppress and publish tp;
    end if
  end for
end if

- Set_{kc}={(([22 − 24], $university$), ([31 − 39], $staff$),([44 − 46], $staff$))}
- C_{new}=(<id_n, 45, $academic$>,<id_{n+2} , 26, $non-academic$>)
- $g_{cnew} = ([26-45], staff)$
- $C_{k-best} = ([44-46], staff)$

In this stage, information loss of C_{k-best} is compared with g'_{cnew} information loss. As, The information loss of C_{k-best} is less than g_{cnew} , tuple with id_n is published with C_{k-best} generalization. Then, the expiration time of tuples with id_{n+1} and id_{n+2} are checked. Because they have enough time to process again, they are added to S for another round. The published tuples are shown in Table 2.

index	PID	Age	University_person
1	id_1	22	Bachelor
2	id_2	24	Master
3	id_3	37	Non-Academic
⋮	⋮	⋮	⋮
n	id_n	45	Academic
n+1	$id_n + 1$	26	Non-Academic
n+2	$id_n + 2$	39	PhD

Table 1: University_person

PID	Age	University_person
id_1	[22-24]	Student
id_2	[22-24]	Student
id_3	[15-95]	Person-University
⋮	⋮	⋮
id_n	[44-46]	Staff
id_{n+1}	publish Next Round	publish Next Round
id_{n+2}	publish Next Round	publish Next Round

Table 2: two-anonymized University_person

.

4.2 Proactive Heuristic

In FADS, a new parameter is considered that represented the maximum delay that is tolerable for an application. This parameter is called expiration-time (t_e). To prevent a tuple be published when its expiration-time passed, a simple heuristic $estimated-round-time$ is defined. This parameter is updated in each round of the algorithm. Then Equation 6 is checked for each remaining tuples and if it is true, the tuple is returned to S for another round. Otherwise, it will be suppressed and published urgently.

$$((current\_time - Arrival\_time) + estimated\_round\_time < expiration\_time)$$

In FADS, there is no check for whether a tuple can remain more in the system or not. As a result, some tuples are published after expiration. This issue is violated the real-time condition of a data stream application and also increase cost metric notably. To depict this issue consider table 1 again. In the first round, tuples id_1, id_2,id_3 are chosen and passed to thread 1. According to Psudo code, the value of variables is as follows:

- Set_{tp}={(<id_1, 22, $bachelor$>,<id_2, 24, $master$>, <id_3, 37, $Non-academic$>)}
- $Set_{kc} = \varnothing$
- $C_{new} = (id_1, 22, bachelor, < id_2, 24, master >)$
- $g_{cnew} = ([22-24], Student)$

tuples with identifierid_1,id_2 are generated 2-anonymous cluster C_{new} and will be published with g_{cnew}. The Equation 6 is checked for tuple with id_3. As this condition is false for this tuple, it is suppressed to $< [15-95], University_Person >$ and published immediately. The results are shown in Table 2.

5. SIMULATION ANALYSIS

We simulated our approach to evaluate its performance and compared it with existing algorithms. The authors in [2] claim that their algorithm is much better than CASTLE and FAANST. So we only compare our proposed algorithm with FADS. The algorithms implemented by java with JDK 7.0.11 on Intel Core i5 2.5 GHz with a Windows 7x64 operating system and 2GB of main memory.

In this experiment, we compared the performance of FAST and FADS on the Adult dataset from UCI [2] . The dataset was broadly used in privacy preserving literature, and it is also used by [2]. It has 6 numerical attributes and 8 categorical attributes. The taxonomy trees are defined by [6] is used to specify the range of each numerical attribute and DGH for categorical attributes. Six numerical attributes as age, final-weight, education-number, capital-gain, capital-loss, hours-per-week and four categorical attributes as education, marital-status, work-class and nation, are selected as quasi-identifier attributes. The sensitive attribute is occupation. To simulate duplicated pids in the data stream, 10 % of the records are randomly selected and inserted back into the original dataset. Therefore, the total number of the records for experiments is 33,178. This is exactly the same dataset used in [7].

The algorithm efficiency was verified with varying parameter setting. The dataset size, k-anonymity degree, size of each variable used in the algorithm, expiration time, and the number of threads was analyzed to show the effect of them on the proposed algorithm efficiency. The default values of parameters are depicted in Table 3. To show the performance of our algorithm to handle heavy traffic, each thread was executed on a single core. As the simulation result shows, the proposed algorithm not only is efficient and effective but also is useful to cope with big data streams.

Table 3: Simulation parameters

Parameters	Values
K	5
$\|QID\|$	10
δ	30
T_{kc}	20
# of Threads	5
A	0.02

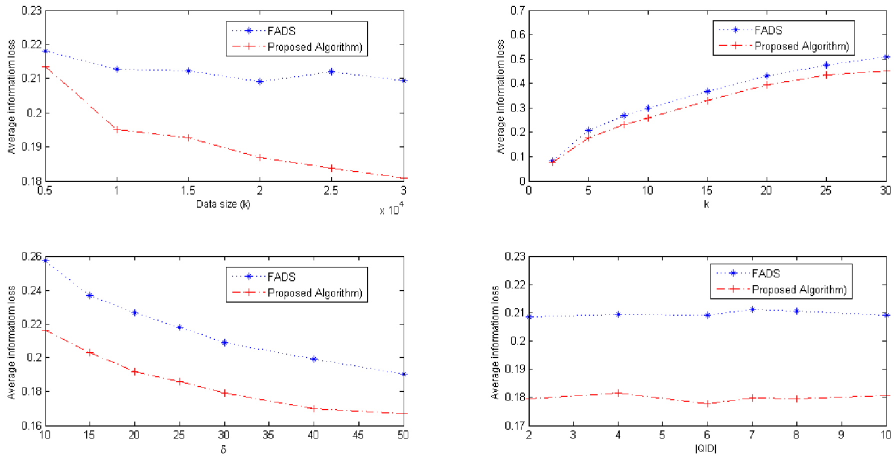

Figure 3: Average information loss varying simulation parameters.

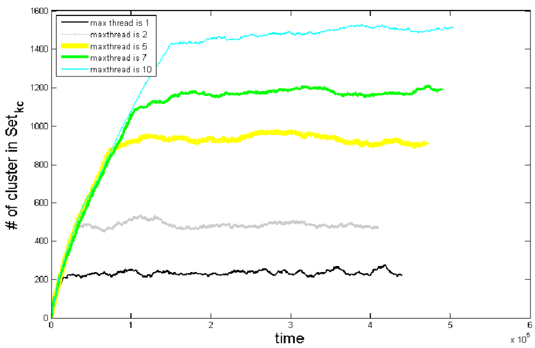

Figure 4: Number of cluster in Set_{kc}.

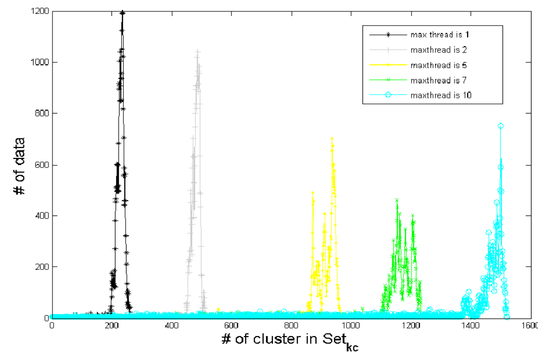

Figure 5: Number of data published varying the size of Set_{kc}.

The average loss of the proposed algorithm and FADS is presented in Figure 3. As the figure shows the proposed algorithm publishes data with less information loss in comparison with FADS. The reason is Set_{KC} in the proposed approach has more entities so the data has more option to select and this would decreases the information loss.

Figure 4 illustrates the size of Set_{KC} during the execution time. The maximum size of Set_{KC} increased by increasing in the number of threads. As a result each data has more clusters to choose.

Figure 5 represents the number of data published varying the number of cluster in Set_{KC}. When the number of thread increases the more data will be publishes with more cluster in Set_{KC}. That is the reason why the proposed model decreases the information loss.

The effect of number of thread in the proposed algorithm is represented in Figure 6. The average execution time is tremendously decreases when the number of threads increases. This make proposed algorithm applicable in real high traffic.

Figure 7 illustrates the cost metric calculated according to Equation 8. The proposed algorithm produces less cost while each packet publishes faster than FADS. This shows that the proposed algorithm is more useful for big data stream when the data should be published with low latency.

The Figure 8 represents the number of round each data participate in the FADS algorithm. It shows there is a significant number of data which are participate in more than 10 round of algorithm (29%). But this is not applicable in many big data stream applications. In FAST the estimation

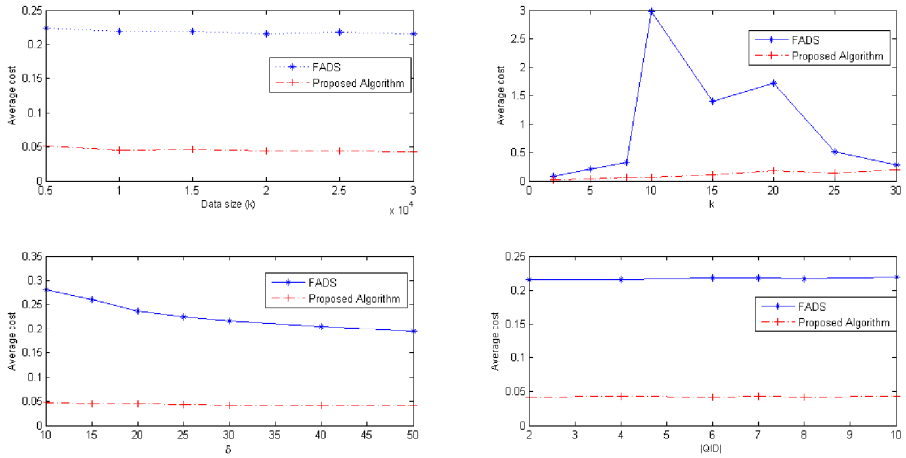

Figure 7: Average cost varying simulation parameters.

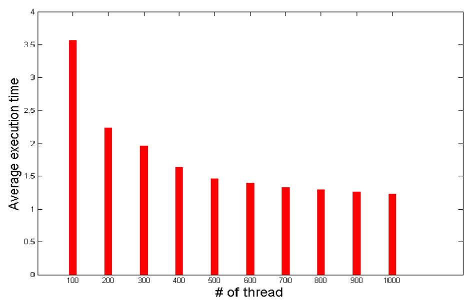

Figure 6: Average execution time varying number of threads.

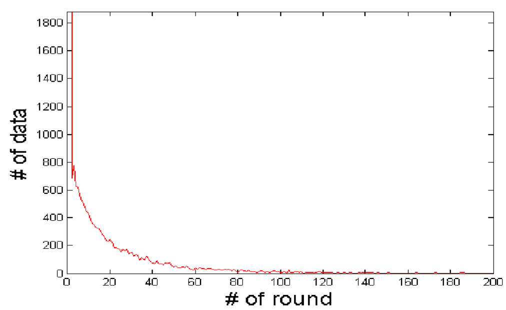

Figure 8: The number of data are published in specific round of FADS.

time parameter can be initialized based on the specification of the applications.

6. CONCLUSION

In this paper, we have presented FAST as a parallel anonymization algorithm to protect privacy of big data stream. A new proactive heuristic is proposed in order to publish data before a specific expiration-time passed. The results of experiments demonstrate the efficiency and effectiveness of FAST for anonymizing big data stream. This algorithm also decreases the information loss and cost metric noticeably. In the future, we plan to design and implement FAST in a distributed cloud-based framework in order to gain cloud computation power and achieve high scalability.

7. REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment, 2003.

[2] C. Blake and C. J. Merz. {UCI} repository of machine learning databases. 1998.

[3] J. Cao, B. Carminati, E. Ferrari, and K.-L. Tan. Castle: Continuously anonymizing data streams. *Dependable and Secure Computing, IEEE Transactions on*, 8(3):337–352, 2011.

[4] C. Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[5] B. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (CSUR)*, 42(4):14, 2010.

[6] B. C. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 205–216. IEEE, 2005.