Asia-pacific Journal of Convergent Research Interchange
Vol.9, No.12, December 31 (2023), pp.21-31
http://dx.doi.org/10.47116/apjcri.2023.12.03

# Proposal of a New Anonymization Algorithm for Real-Time Stream Data

# 실시간 스트림 데이터를 위한 새로운 익명화 알고리즘 제안

Sung Hyun Hong[1], Gyu Sung Lee[2], Dong le Kim[3], Soon Seok Kim[4]

홍성현[1], 이규성[2], 김동례[3], 김순석[4]

[1] *Senior Researcher, Easycerti Inc., Korea, shhong@easycerti.com*
[2] *Executive Director, Easycerti Inc., Korea, kslee@easycerti.com*
[3] *Researcher, Easycerti Inc., Korea, drkim@easycerti.com*
[4] *Professor, Department of AI Convergence Security, Halla University, Korea, sskim@halla.ac.kr*

*Corresponding author: Soon Seok Kim*

**Abstract**: Real-time stream data refers to data which collected in real time such as personal vital signs information collected from various PoCs (Point of Care, point-of-care medical equipment) in hospitals, real-time crime report information, and online sales transaction. We propose a new anonymization algorithm for privacy protection in these real-time collected stream data. The proposed algorithm significantly improved performance in four aspects compared to the UBDSA algorithm proposed by Ugur and Osman. First, after forming a generalization tree, precompute was performed to measure information loss in advance. Second, whenever each transaction data (record) is entered, quasi-identifier columns and non-identifier columns are separated and stored through slicing, and serial numbers are attached to each slice separated at the time of separation for combine before publishing, That is, when clustering for each transaction, only quasi-identifier columns are stored and generalized to reduce storage space and improve calculation performance. In particular, it is more efficient for large numbers of columns or large-capacity data. Third, the performance in the cluster assignment (AssignCluster) step was improved as follows. First, from the initial cluster assignment process below the delay threshold (the maximum allowable number of transaction records from input to publishing), clustering was performed in consideration of information loss, rather than assigning to separate clusters, unlike existing algorithms. Finally, performance was improved in generalization and publishing (Publish stage). First, after forming a generalization tree, precompute was performed to measure information loss in advance. Second, whenever each transaction data (record) is entered, quasi-identifier columns and non-identifier columns are separated and stored through slicing, and serial numbers are attached to each slice separated at the time of separation for combine before publishing, That is, when clustering for each transaction, only quasi-identifier columns are stored and generalized to reduce storage space and improve calculation performance. In particular, it is more efficient for large numbers of columns or large-capacity data. Third, the performance in the cluster assignment (AssignCluster) step was improved as follows. First, from the initial cluster assignment process below the delay threshold (the maximum allowable number of transaction records from input to publishing), clustering was performed in consideration of information loss, rather than assigning to

Received: August 04, 2023; 1st Review Result: September 08, 2023; 2nd Review Result: October 11, 2023
Accepted: November 25, 2023

separate clusters, unlike existing UBDSA algorithm. Finally, performance was improved in generalization and publishing (Publish stage). The partitioning process is not performed for clusters with more than 2k tuples in the cluster, and unlike the existing data quality improvement claims, data quality can be good without partitioning and safety is higher. It improves performance by not running the process. If the number of tuples in a cluster is less than k, after searching for other movable clusters, in the process of searching and assigning, possible clusters are found and assigned based on the minimum distance rather than random assignment. Contrary to the previous argument, if the minimum value of k is guaranteed to avoid counterattack due to information leakage, it is reasonable to assign it to the cluster with the minimum distance in terms of usefulness. And if the number of tuples in a cluster is less than k previously, if no possible cluster is found after searching for other movable clusters, or if the number of tuples in a non-anonymized cluster is smaller than the medium-sized cluster, then we can reduce execution time because we choose only deletion rather than existing generalization of the top-level or deletion.

**Keywords**: Real-time Stream Data, Privacy, Anonymization, Implementation

**요약**: 실시간 스트림 데이터라 함은 실시간 범죄신고정보, 온라인 판매거래정보, 병원 내 환자 모니터링 장치 등 각종 PoC(Point of Care, 현장 진료형 의료장비)들로부터 실시간으로 수집되는 개인생체신호 등의 데이터를 말한다. 우리는 이러한 실시간 수집 스트림 데이터에서의 프라이버시 보호를 위한 새로운 익명처리 알고리즘을 제안하고자 한다. 제안 알고리즘은 기존 Ugur와 Osman이 제안한 UBDSA 알고리즘보다 크게 4가지 측면에서 성능을 개선하였다. 첫째, 일반화 트리를 형성한 다음 정보손실을 미리 측정할 수 있도록 사전 수행과정(Precompute)을 두었다. 둘째, 각 트랜잭션 데이터(레코드)가 입력될 때마다 슬라이싱을 통하여 준식별자 컬럼들과 아닌 컬럼들을 분리하여 각각 저장하였으며, 아울러 게시전 조합(combine)을 위하여 분리시 분리된 각 슬라이스에 일련번호를 붙이도록 하였다. 즉, 각 트랜잭션 별로 클러스터링시 준식별자 컬럼들만 저장하여 일반화함으로써 저장공간을 줄이고 계산 성능을 개선하였으며, 특히 이것은 컬럼수가 많거나 혹은 대용량 데이터에 대해 보다 효율적이다. 셋째, 클러스터 할당(AssignCluster) 단계에서 성능을 아래와 같이 개선하였다. 먼저 지연 임계값(입력 후 게시될때 까지의 최대 허용 가능 트랜잭션 레코드 수)이하 초기 클러스터 할당 과정에서부터 기존 알고리즘과 달리 각각 별도의 클러스터에 배정하지 않고 정보손실을 고려하여 클러스터링을 수행하였다. 끝으로 일반화 및 게시(Publish 단계)에서 성능을 개선하였다. 기존과 달리 클러스터 내 튜플의수가 2k 이상인 클러스터에 대해 분할 과정을 미수행하며 기존 주장인 데이터 품질 개선과는 달리 굳이 분할을 하지 않더라도 그 자체로도 데이터 품질이 좋을 수도 있으며, 안전성이 보다 높고 과정을 수행하지 않음으로 인해 성능도 향상된다. 또한 기존 UBDSA 알고리즘과 달리 클러스터 내 튜플의수가 k보다 적은 경우 이동 가능한 타 클러스터를 탐색 후 가능 클러스터를 찾아 이를 배정하는 과정에서 랜덤 배정이 아닌 최소거리 기반 배정을 수행하였다. 이는 기존 주장인 정보유출로 인한 반대 공격을 피하는 것과 달리 최소 k값이 보장될 경우 유용성 차원에서 최소거리인 클러스터로 배정이 타당하다. 그리고 앞서 클러스터 내 튜플의 수가 k보다 적은 경우 이동 가능한 타 클러스터를 탐색 후 가능 클러스터를 못찾았거나 혹은 익명화되지않은 클러스터 내 튜플수가 중앙값 클러스터 크기보다 작은 경우 기존 삭제 혹은 최상위 일반화 대신 제안 알고리즘은 삭제를 수행함으로서 수행 시간을 줄일 수 있다.

**핵심어**: 실시간 스트림 데이터, 프라이버시, 익명화, 구현

# 1. 서론

실시간 수집 스트림 데이터를 이용한 각종 분석 및 활용은 현재 다양한 분야에 응용되고 있다. 예로 주민이 신고한 강력범죄(2022년 기준 우리나라 24,954건, 국가통계포털[1]) 등을 실시간으로 분석하여 짧은 시간에 범죄자나 용의자를 발견하거나 미래 범죄 혹은 재난 발생에 대한 예측을 가능하게 한다거나 캐나다의 Questrade.com과 같은 온라인 거래 회사는 고객이 발행한 수십만건의 온라인 입찰을 매일 수집하고 있는데, 이때 입찰자의 입찰정보는 매번 실시간 스트림 형태로 각 회사 주식에 대한 통계 정보를 찾아 실시간으로 게시하기 위하여 데이터 마이닝 알고리즘을 이용, 거래에서 생성된 데이터 스트림을 처리하고 있다[2]. 그밖에 병원 내 환자 모니터링 장치 등 각종 PoC장비들로부터 수집한 개인생체신호들을 HL7[3]이나 FHIR[4] 등 표준 메시징 형식으로 변환하여 실시간으로 질병 징후를 탐지 분석하는 등에 활용되고 있다.

앞서 살펴본 사례에서 범죄자나 용의자, 온라인 입찰 참가자, 그리고 환자 등은 개인정보로서 프라이버시를 위해 보호될 필요성이 있다. 우리는 이러한 응용들에서 수집된 데이터로부터 정보 주체의 프라이버시를 보호하기 위한 익명화 알고리즘을 새롭게 제안하고 이를 설계 및 구현하고자 한다.

이러한 실시간 수집 스트림 데이터의 익명화와 관련하여 그동안 많은 연구들[5-11]이 있어왔으며 현재까지의 연구 중 가장 개선된 방법은 UBDSA[11] 알고리즘이다. UBDSA 알고리즘 이전의 연구들은 버퍼 내로 들어오는 레코드들의 평균 지연(δ) 최소화를 목적으로 개발된 것이 아니라 최대 지연 시간을 임계값으로 정해두고 정보손실을 최소화하는 방향으로 접근해왔다. 데이터 에이징이란 데이터 수집과 공유 사이의 시간 지연을 뜻하며, 데이터 유틸리티란 데이터 품질과 데이터 에이징 모두의 기능 즉, 시간 지연과 정보손실을 모두 고려한 것이다. 따라서 데이터 유틸리티는 평균 지연 최소화와 데이터 익명화 후 즉, 공개시 데이터 품질(Data quality)의 합으로 확인할 수 있다. 이때 지연 임계값 δ의 증가는 평균 지연 증가와 더 많은 레코드의 유입으로 정보 손실 감소를 유발한다. 따라서 본 논문에서는 상기 둘 모두를 고려하여 기존 UBDSA 알고리즘보다 성능이 우수한 효율적인 알고리즘을 개발하고 이를 구현하였다. 본 논문은 제안 알고리즘 보다 이를 어떻게 설계하고 구현하였는지에 초점을 맞춘다.

본 논문의 2장에서는 관련 연구로 기존에 제안된 방법들에 대해 살펴보고 3장에서 제안 알고리즘을 기술한다. 4장에서 제안 알고리즘의 차별성과 우수성에 대해 분석한 다음 5장을 끝으로 결론을 맺고자 한다.

# 2. 관련 연구

실시간 수집 스트림 데이터의 익명화와 관련하여 기존에 제안된 방법 및 특징은 아래 [표 1]과 같다. 아울러 가장 최근에 제안된 UBDSA[11] 알고리즘의 슈도코드는 [그림 1]과 같다. [그림 1]에서 S는 입력 레코드, δ는 최대지연 임계값, k는 k-익명성 프라이버시 보호 모델[12]에서의 최소 k값, w는 윈도우 사이즈 매개변수(QI-그룹 수), stepsize는 단계크기 매개변수(현 δc에 대한 단계별 증감치)를 말한다. 6라인의 AssignCluster 함수는 NonAnonClusters중 하나로 트랜잭션 $t_j$를 할당하거나 단지 $t_j$만을 담은 클러스터를 생성하여 이것을 기존 익명화 안된 클러스터에 합치는 기능을 수행한다.

[표 1] 실시간 수집 스트림 데이터 익명화에 대한 기존 제안 방법 및 특징 요약

[Table 1] Summary of Existing Methods and Features for Real-time Collection Stream Data Anonymization

| 기존 제안 방법 | 주요 특징 |
|---|---|
| SWAF(Wang et al. 2007)[5] & SKY(Li et al. 2008)[6] | · 데이터 스트림 익명화에 대한 가장 초기 제안<br>· 최대 지연 제약 조건에서 튜플은 버퍼에 보관되고 k-익명화 프로세스는 정보 손실 최소화를 목적으로 유사 식별자 속성을 일반화<br>· 일반화는 각 속성 전문화 트리에 대한 하향식 검색을 기반으로 함 |
| CASTLE(Cao et al. 2011)[7] | · 클러스터링기반 접근, CASTLE(Continuously Anonymizing Streaming data via adaptive cLustEring)<br>· k-익명성 프라이버시 원칙에 따라 데이터 스트림을 익명화<br>· 확장(Enlargement)이라는 튜플 대 클러스터 거리 메트릭을 사용하여 튜플을 익명성 그룹으로 클러스터링<br>· 확장 메트릭은 튜플이 클러스터에 할당되기 전과 후의 클러스터 일반화 양 간의 차이를 측정<br>- 클러스터링의 목적은 일반화로 인한 데이터 왜곡이 가능한 한 작도록 유사한 튜플을 동일한 유사 식별자 그룹에 모으는 것임<br>- 일정에 따라 가장 오래된 튜플이 포함된 클러스터는 버퍼가 가득 차면 해제됨<br>- 튜플을 하나의 큰 클러스터로 누적시킨 다음 과도하게 일반화하는 경향이 있음<br>- 따라서 일반화되지 않은 다른 클러스터에는 제한된 수의 튜플이 포함됨<br>- 확장 메트릭은 도착하는 튜플과 후보 클러스터 간의 유사성을 올바르게 측정하지 못하는 경우가 있음 |
| FAANS(Zakerzadeh 및 Osborn 2011)[8] | · 클러스터링 기반 접근<br>· 숫자 유사 식별자 속성에만 해당되며 범주 속성을 처리할 수 없음<br>· k-평균 클러스터링을 사용 |
| FADS(Guo 및 Zhang 2013)[9] | · 클러스터링 기반 접근<br>· 모든 클러스터 크기를 $k$로 수정함<br>· FAANST와 같은 버퍼창을 사용하지만 범주 및 숫자 속성을 모두 지원<br>· 들어오는 튜플이 고정된 길이의 버퍼 내에 누적됨<br>· 버퍼가 가득 차면 가장 오래된 튜플이 시드 멤버의 역할을 하는 튜플 클러스터를 만드는 익명화 단계가 시작됨<br>· 간단하고 직관적인 설계를 통해 FADS는 매우 효율적이면서도 효과적인 (즉, 평균 정보 손실 측면에서) 방법임<br>· 평균 정보 손실 측면에서 CASTLE, B-CASTLE 및 FAANST보다 더 나은 결과를 제공한다고 주장함. 그러나 버퍼에서 튜플의 평균 지연이 너무 높음<br>· 예를 들어 실험에서 FADS의 평균 지연이 CASTLE보다 평균 40% 더 높음 |
| Kim et al (2014)[10] | · 지연없는 익명화 방법 제안, 그러나 이 방법은 l- 다양성에 적용 할 수 있으며 k-익명성에는 적용되지 않음 |
| UBDSA(Ugur & Osman 2020)[11] | · 클러스터링기반 익명화, UBDSA(Utility Based Approach for Data Stream Anonymization)<br>· 숫자 및 범주 유사 식별자를 모두 처리<br>· 평균지연과 정보손실 모두를 고려<br>· CASTLE 계열 단점 개선<br>- 데이터 유틸리티 (데이터 품질 및 데이터 노화)를 처리<br>- CAIL(Cardinality Aware Information Loss, Cluster Assignment Distance Metric)이라는 새로운 클러스터 할당 거리 메트릭을 개발. 기존 Enlargement 방식의 단점과 성능 개선 |

8라인은 δc내로 지연을 막기위해 가장 오래된 트랜잭션 레코드 to를 찾아 현 δc 대비 대기시간 to를 체크하며 10라인의 Publish 함수는 각각의 익명화된 클러스터는 처음에 최소 일반화되어 프로토타입 형태로 AnonClusters에 있다가 Publish 프로시져를 통해 AnonClusters 및 NonAnonClusters를 활용하여 시드 튜플 $t_o$에 대한 좋은 준식별자 그룹을 찾거나 만든 후 공개한다. 11라인의 UpdateDelta 함수는 δc의 값을 업데이트하며, AnonClusters 내에서 최근에 익명화된 2w QI 그룹의 내용에 따라 단계적으로 증가하거나 감소한다. 이때, 윈도우 사이즈 매개 변수 w 및 단계 크기 매개 변수 stepsize는 함께 제공되어 평균 정보 손실과 평균 지연 간의 균형을 조정하는 역할을 수행한다. 12라인은 최근에 익명화된 클러스터들을 AnonClusters로 이동하는 역할을 수행한다.

**Algorithm 1.** UBDSA algorithm

**Input :** *S, δ, k, ω, stepsize*
**Output :** *S'*
1: *AnonClusters*← ∅
2: *NonAnonClusters*← ∅
3: $j \leftarrow 1$
4: $\delta_c \leftarrow \delta$
5: **while** $S_j$ has next tuple $t_j$ **do**
6: ***AssignCluster***(*$t_j$, NonAnonClusters*)
7: *UnPublishedTuples*← $S_j \setminus S_j$'
8: $t_o$← oldest tuple from *UnPublishedTuples*
9: **while** $j - o \geq \delta_c$ **do**
10: ***Publish***($t_0$, *AnonClusters*, *NonAnonClusters*)
11: ***UpdateDelta***($\delta_c$, *δ*, *k*, *ω, stepsize*, *AnonClusters*)
12: Shift *AnonClusters*
13: $t_0$← oldest tuple from *UnPublishedTuples*
14: $j \leftarrow j+1$

**Algorithm 2.** Assigncluster procedure

**Input :** $t_j$ , *NonAnonClusters*
**Output :** updated *NonAnonClusters*
1: $C_{minDist} \leftarrow argmin_c\{CAIDistance(t{\cdot}c)|c \in NonAnonCluster\}$
2: $C_{minIL} \leftarrow \{c|\ c \in C_{minDist}$ and $IL(\{t_j\} \cup c) \leq \tau\}$
3: **if** $C_{minIL} \neq \emptyset$ **then**
4: $C_{cand} \leftarrow argmin_c\{||c||c \in C_{minIL}\}$
5: $C_{pick}$← A random cluster from $C_{cand}$
6: $C_{pick} \leftarrow C_{pick} \cup \{t_j\}$
7: **else**
8: **if** $|NonAnonClusters| < \beta$ **then**
9: $C_{pick}$← A random cluster from $C_{minDist}$
10: $C_{pick} \leftarrow C_{pick} \cup \{t_j\}$
11: **else**
12: $NonAnonClusters \leftarrow NonAnonClusters \cup \{t_j\}$

**Algorithm 3. Publish procedure**

**Input** : $t_o$, *AnonClusters*, *NonAnonClusters*
**Output** : updated *AnonClusters*, *NonAnonClusters*
1: $C_{t_0} \leftarrow \{c | c \in NonAnonClusters \text{ and } t_0 \in c\}$
2: **if** $|C_{t_0}| \geq k$ **then**
3: *NonAnonClusters* ← *NonAnonClusters*$\setminus\{C_{t_0}\}$
4: **if** $|C_{t_0}| \geq 2k$ then
5: $C \leftarrow$ ***splitCluster***$(C_{t_0})$
6: **for each** $C_i \in C$ **do**
7: ***outputCluster***$(C_i)$
8: **else**
9: ***outputCluster***$(C_{t_0})$
10: **else**
11: $C \leftarrow$ ***findFittingCluster***$(t_o, AnonClusters)$
12: **if** $C \neq \emptyset$ **then**
13: $C_{pick} \leftarrow$ A random cluster from $C$
14: ***outputTuple***(***generalize***$(t_0, C_{pick})$)
15: $C_{t_0} \leftarrow C_{t_0} \setminus \{t_0\}$
16: **else**
17: *NonAnonT* ← $\{t : c \in NonAnonClusters \text{ and } t \in c\}$
18: small $C \leftarrow \{c : c \in NonAnonClusters \text{ and } |c| \langle |C_{t_0}|\}$
19: **if** $|NonAnonT| < k$ or $|smallC| \leq |NonAnonClusters|/2$ **then**
20: $C_{t_0} \leftarrow C_{t_0} \setminus \{t_0\}$
21: ***outputTuple***(***generalize***$(t_0, *)$)
22: **else**
23: ***mergeCluster***$(C_{t_0}, NonAnonClusters)$
24: ***outputCluster***$(C_{t_0})$
23: *NonAnonClusters* ← *NonAnonClusters*$\setminus\{C_{t_0}\}$

[그림 1] 기존 UBDSA[7] 알고리즘의 슈도코드

[Fig. 1] Pseudo Codes of Existing UBDSA[7] Algorithm

본 논문에서는 기존 UBDSA[7] 방법을 기본적으로 채택하되 기존 제안 방법인 k-익명성 모델에 기반하여 기존 알고리즘의 과정 개선을 통하여 처리 성능을 향상시키는 것을 목표로 한다.

## 3. 제안 알고리즘

제안 알고리즘은 크게 4가지 단계로 구성되어 있으며 아래와 같다. 한편 [그림 2]는 제안 알고리즘의 슈도코드이다.

[단계 1] 사전수행(Precompute)
(1-1) 실시간 입력 데이터에 대해 준식별자 지정
(1-2) 지정된 준식별자 속성들에 대해 일반화 트리 구성
(1-3) 정보손실 계산 성능 향상을 위하여 일반화 트리 내 모든 노드들에 대하여 leaves 노드수를 미리 계산하여 노드와 함께 저장
[단계 2] 클러스터 할당(AssignCluster)
(2-1) [Slicing] 새 트랜잭션 입력시Slicing 수행

**Algorithm 1.** ***Precompute*** **Procedure**

1: Assign identifiers and quasi-identifiers to real-time input data
2: Construct a generalization tree for specified quasi-identifier attributes
3: In order to improve the calculation performance of the information loss, the number of leaves nodes is calculated in advance for all nodes in the generalized tree and stored together with the nodes.

**Algorithm 2. The Proposed Algorithm**

**Input :** $S$, $\delta$, $k$, $\omega$, *stepsize*
**Output :** $S'$
1: *AnonClusters*← ∅
2: *NonAnonClusters*← ∅
3: $j \leftarrow 1$
4: $\delta_c \leftarrow \delta$
5: **while** has next tuple **do**
6: ***AssignCluster*** $\{t_j$, *NonAnonClusters*$\}$
7: *UnPublishedTuples* ← $S_j \setminus S_j'$
8: $t_o$← oldest tuple from *UnPublishedTuples*
9: **while** $j - o \geq \delta_c$ **do**
10: ***Publish***$(t_0$, *AnonClusters*, *NonAnonClusters*$)$
11: ***UpdateDelta***$(\delta_0, \delta, k, \omega$, *stepsize*, *AnonClusters*$)$
12: Shift *AnonClusters*
13: $t_0$← oldest tuple from *UnPublishedTuples*
14: $j \leftarrow j + 1$

**Algorithm 3.** ***AssignCluster*** **Procedure**

**Input :** $t_j$, *NonAnonClusters*
**Output :** updated *NonAnonClusters*
1: Delete identifiers, ID
2. Divide the QI and non-QI columns (SA, NSA) into two and keep the non-QI columns separately by attaching a serial number to each, For the remaining quasi-identifier columns,
3. **if** $|$*NonAnonClusters*$| < \beta$ **then**
4: **if** first tuple $t_1$ **then**
5: *NonAnonClusters* ← *NonAnonClusters* $\cup$ $\{t_1\}$
6: **else if** the tuple $t_i$ has the same ancestor in the generalization tree as the tuples in the previously allocated cluster $C_i$ **then**
7: $C_{pick}$ ← cluster $C_i$
8: $C_{pick} \leftarrow C_{pick} \cup \{t_i\}$
9: **else**
10: *NonAnonClusters* ← *NonAnonClusters* $\cup$ $\{t_j\}$
11: **else**
12: $C_{minDist} \leftarrow argmin_c\{$*CAILDistance*$(t \cdot c) \mid c \in$ *NonAnonClusters*$\}$ based *Precompute* procedure
13: $C_{minIL} \leftarrow \{c \mid c \in C_{minDist}$ and $IL(\{t_j\} \cup c) \leq \tau\}$
14: $C_{cand} \leftarrow argmin_c\{|c| c \in C_{minIL}\}$
15: $C_{pick}$ ← A random cluster from $C_{cand}$
16: $C_{pick} \leftarrow C_{pick} \cup \{t_j\}$

**Algorithm 4.** ***Publish*** **Procedure**

**Input :** $t_0$, *AnonClusters*, *NonAnonClusters*
**Output :** updated *AnonClusters*, *NonAnonClusters*

```
1: c_{t_0} ← {c| c ∈ NonAnonClusters and t_0 ∈ c }
2: if |c_{t_0}| ≥ k then
3:         NonAnonClusters ← NonAnonClusters\{c_{t_0}}
4: for each c_i ∈ C do
5:         outputCluster(c_i)
6:         else
7:                 outputCluster(c_{t_0})
8: else
9:         C ← findFittingClusters(t_0, AnonClusters)
10:        if C = ∅ then
11:        C_pick ← {c| c ∈ C_minDist} from C
12:        outputTuple(generalize(t_0, C_pick))
13:        c_{t_0} ← c_{t_0}\{t_0}
14:        else
15:                NonAnonT ← {t : c ∈ NonAnonClusters and t ∈ c }
16:                small C ← {c : c ∈ NonAnonClusters and |c|⟨|c_{t_0}|}
17:                if |NonAnonT|⟨ k or |smallC| ≤ |NonAnonClusters|/2 then
18:                c_{t_0} ← c_{t_0}\{t_0}
19:                outputTuple(suppression(t_0))
20:                else
21:                        mergeCluster(c_{t_0}, NonAnonClusters)
22:                        outputCluster(c_{t_0})
23:                        NonAnonClusters ← NonAnonClusters\{c_{t_0}}
```

**Algorithm 5.** ***UpdataDelta*** **Procedure**

```
Input : δ_c, δ, k, ω', stepsize, AnonClusters
Output : updated δ_c
1: check ω'
2: if ω' ≥ 2ω then
3:     If C'_minDist⟩ C_minDist then
4:     δ_c= δ_c+ stepsize
5:     else
6:         δ_c= δ_c- stepsize
```

[그림 2] 제안 알고리즘의 슈도코드

[Fig. 2] Pseudo Codes of Proposed Algorithm

(2-1-1) ID삭제 후 QI와 아닌 컬럼들(SA, NSA)을 둘로 쪼개고 각각에 일련번호를 붙여 아닌 컬럼들은 별도 보관하고 준식별자 컬럼들만 아래 2.단계로 이동

(2-2) [Assign Clustering] 실시간 입력 트랜잭션 데이터(튜플(레코드)단위)를 각각의 클러스터에 할당

(2-2-1) 할당된 클러스터의 최대 개수를 β라 할때,입력 튜플수가 β개 이내일 경우, 이때 만일 준식별자가 일반화 트리에서 바로 위 공통 조상일 경우 동일 클러스터에 할당

(2-2-2) β개를 초과할 경우, 정보손실을 계산하여 제일 적은 기존 클러스터에 배정

[단계 3] 일반화 및 데이터 공개(Publish)

(3-1) [Generalization] 각클러스터별로 저장된 튜플들에 대해 일반화 수행(허용 최대 지연시간 도래시 수행됨), 이때 만일 지연시간(입력 후 게시될때까지의최대 허용 가능 트랜잭션 레코드 수 δc)이 가장 오래된 튜플(to)이 속한 클러스터(cto) 내 총 레코드의 개수가

(3-1-1) k(-익명성모델)보다 크거나 같은 경우, 모든 QI 속성에 대한 최소 공통 조상으로 일반화

(3-1-2) k보다 작고 이동할 다른 가능 일반화 클러스터가 있을 경우, 해당 to를 함께 일반화

(3-1-3) k보다 작고 이동할 다른 가능 일반화 클러스터가 없을 경우, 삭제 혹은 정보손실이 제일 적은 클러스터와 병합하고 일반화

(3-2) [Release] 일반화된 클러스터별로 별도 보관된 준식별자가 없는 컬럼셋과 병합 후 게시

(3-2-1) 일반화된 클러스터별로 [Step 2] 1. Slicing 단계에서 분리된 비준식별자 컬럼들과 병합 후 게시

[단계 4] 지연시간 조정(UpdateDelta)

(4-1) [UpdateDelta] 정보손실을 고려하여 지연시간 조정

(4-1-1) 게시전과 게시 후 각각의 평균 정보손실을 측정, 이전보다 손실이 커졌을 경우 지연시간(δc)를 늘리고(미리 정한 stepsize만큼) 그 반대인 경우는 줄임, 이때 본 함수 호출 시점은 미리 정함(기존 UBDSA[7] 알고리즘의 경우 2ω로 이때 ω는 QI-그룹수를 말함)

## 4. 제안 알고리즘의 분석

제안하는 방법은 기본적으로 앞서 UBDSA[11]의 방법과 큰 틀에서 기본적인 맥락은 동일하지만 세부 수행과정과 성능 측면에서 아래와 같은 4가지의 주요 차이점이 있다.

첫째, 기존 [11]에서와 달리 별도 사전 수행과정(Precompute)을 통해 정보손실을 미리 계산함으로써 성능을 향상시켰다. 둘째, 각 트랜잭션 데이터(레코드)가 입력될 때마다 슬라이싱을 통하여 준식별자 컬럼들과 아닌 컬럼들을 분리하여 각각 저장하고 아울러 게시 전 결합을 위하여 분리시 분리된 각 슬라이스에 일련번호를 붙였으며 세부적으로는 a) 각 트랜잭션별로 클러스터링시 준식별자 컬럼들만 저장하여 일반화함으로써 저장공간을 줄이고 계산 성능을 개선하고 b) 특히 컬럼수가 많거나 혹은 대용량 데이터에 대해 보다 효율적이다. 셋째, AssignCluster 과정에서 성능을 개선하였다. 세부적으로는 a) 지연 임계값(입력 후 게시될 때까지의 최대 허용 가능 트랜잭션 레코드 수)이하 초기 클러스터 할당과정에서부터 기존 알고리즘과 달리 각각 별도의 클러스터에 배정하지 않고 정보손실을 고려하여 클러스터링을 수행하였다. 네번째로 Publish 과정에서 성능을 개선하였다. 세부적으로는 a) 기존과 달리 클러스터 내 튜플의 수가 2k 이상인 클러스터에 대해 분할 과정 미수행, 즉 기존 [11]에서의 데이터 품질 개선과는 달리 굳이 분할을 하지 않더라도 그 자체로도 데이터 품질이 좋을 수도 있으며, 안전성이 보다 높고 과정을 수행하지 않음으로 인해 성능도 향상된다. b) 기존과 달리 클러스터 내 튜플의 수가 k보다 적은 경우 이동 가능한 타 클러스터를 탐색 후 가능 클러스터를 찾아 이를 배정하는 과정에서 랜덤 배정이 아닌 최소거리 기반 배정 수행, 즉 기존 [11]에서 정보유출로 인한 반대 공격을 피하는 것과 달리 최소 k값이 보장될 경우 유용성 차원에서 최소거리인 클러스터로 배정이 타당하다. c) 앞서 b)에서 클러스터 내 튜플의 수가 k보다 적은 경우 이동 가능한 타 클러스터를 탐색 후 가능 클러스터를 못찾았거나 혹은 익명화되지 않은 클러스터 내 튜플수가 중앙값 클러스터 크기보다 작은 경우 삭제 혹은 최상화 일반화 보다 삭제를 수행한다.

제안하는 방법과 기존에 제안된 방법들에 대한 성능을 비교하면 아래 [표 2]와 같다.

[표 2] 제안하는 방법과 기존 방법들과의 성능 비교

[Table 1] Performance Comparison Between the Proposed Method and Existing Methods

X : 미지원, ○ : 지원, ◒ : ○보다 우수, ◕ : ◒보다 우수, ● : ◕보다 우수

| 구분 | 클러스터링기반 | 버퍼에 머무는 최대(평균) 지연 시간 | 정보손실 | 숫자 속성 | 범주속성 | k-익명성 |
|---|---|---|---|---|---|---|
| SWAF[5] & SKY[6] | X | ○(최대) | ○ | ○ | | ○ |
| CASTLE[7] | ○ | ◒(최대) | ◒ | ○ | ○ | ○ |
| FAANS[8] | ○ | ○(최대) | ○ | ○ | X | ○ |
| FADS[9] | ○ | ○(최대) | ● | ○ | ○ | ○ |
| KIM[10] | ○ | ●(최대) | ○ | ○ | ○ | X |
| UBDSA[11] | ○ | ◕(평균) | ◕ | ○ | ○ | ○ |
| 제안하는 방법 | ○ | ●(평균) | ● | ○ | ○ | ○ |

## 5. 결론 및 향후 연구 방향

우리는 지금까지 실시간 수집 정형데이터를 대상으로 한 익명처리 방법에 대하여 살펴보았다. 제안하는 방법은 기존 UBDSA[11] 방법에 비하여 평균 지연 시간은 거의 동일하면서도 기존 알고리즘의 처리 성능을 4가지 측면(Precomputing 과정 추가, 컬럼 슬라이싱, AssignCluster, 그리고 Publish과정에서의 성능 개선)에서 개선하였다. 향후 연구방향으로 제안 알고리즘의 성능이 기존에 제안된 UBDSA 알고리즘보다 우수하다는 것을 검증하기 위해 기존 UBDSA 알고리즘을 구현하여 제안 알고리즘과의 성능을 비교하려 한다. 아울러 제안 알고리즘에는 기존 k-익명성 모델이 적용되었지만 이에 l-다양성 프라이버시 모델[13]을 추가하여 안전성을 더욱 개선하고자 한다.

## 6. 감사의 글

이 논문은 2023년도 정부(개인정보보호위원회)의 재원으로 한국인터넷진흥원의 지원을 받아 수행된 연구임 (No. 1781000005, 반정형 트랜잭션 및 실시간 수집 정형 데이터에서의 개인정보 가명,익명처리 기술 개발)

## References

[1] https://kosis.kr/index/index.do, Oct 06 (2023)

[2] https://www.questrade.com/pricing/self-directed-commissions-plans-fees, Oct 05 (2023)

[3] https://www.hl7.org/, Oct 05 (2023)

[4] https://www.hl7.org/fhir/, Oct 05 (2023)

[5] W. Wang, J. Li, C. Ai, Y. Li, Privacy protection on sliding window of data streams, International conference on collaborative computing: networking, applications and worksharing (CollaborateCom 2007), IEEE, pp.213-221, (2007)
DOI: https://doi.org/10.1109/COLCOM.2007.4553832

[6] J. Li, B. C. Ooi, W. Wang, Anonymizing streaming data for privacy protection, IEEE 24th International conference on data engineering (ICDE 2008), IEEE, pp.1367-1369, (2008)
DOI: https://doi.org/10.1109/ICDE.2008.4497558

[7] J. Cao, B. Carminati, E. Ferrari, K. L. Tan, Castle: continuously anonymizing data streams, IEEE Transactions on Dependable and Secure Computing, (2011), Vol.8, No.3, pp.337352.
DOI: https://doi.org/10.1109/TDSC.2009.47

[8] H. Zakerzadeh, S. L. Osborn, Faanst: fast anonymizing algorithm for numerical streaming data, Data privacy management and autonomous spontaneous security, Springer, pp.36-50, (2011)
DOI: https://doi.org/10.1007/978-3-642-19348-4_4

[9] K. Guo, Q. Zhang, Fast clustering-based anonymization approaches with time constraints for data streams, Knowledge-Based Systems, (2013), Vol.46, pp.95-108.
DOI: https://doi.org/10.1016/j.knosys.2013.03.007

[10] S. Kim, M. K. Sung, Y. D. Chung, A framework to preserve the privacy of electronic health data streams, Journal of Biomedical Informatics, (2014), Vol.50, pp. 95-106.
DOI: https://doi.org/10.1016/j.jbi.2014.03.015

[11] Ugur Sopaoglu, Osman Abul, A utility based approach for data stream anonymization, Journal of Intelligent Information Systems, (2020), Vol.54, pp.605-631.
DOI: https://doi.org/10.1007/s10844-019-00577-6

[12] L. Sweeney, k-anonymity: A model for protecting privacy, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, (2002), Vol.10, No.3, pp.557-570.
DOI: https://doi.org/10.1142/S0218488502001648

[13] A. Machanavajjhala, D. Kifer, J. Gehrke, Venkitasubramaniam M, l-diversity: Privacy beyond k-anonymity, ACM Transactions on Knowledge Discovery from Data (TKDD), (2007), Vol.1, No.1, pp.3-14.
DOI: https://doi.org/10.1145/1217299.1217302

 