Optimization and Security of Continuous Anonymizing Data Stream

S. Nasira Tabassum

Department of SE, Nizam Institute of Engineering and Technology, Deshmukhi, Nalgonda, (A.P.), India

Summary:
The characteristic of data stream is that it has a huge size and its data change continually, which needs to be responded quickly, since the times of query is limited. The continuous query and data stream approximate query model are introduced in this paper. Then, the query optimization of data stream and traditional database are compared such as k-anonymity methods, are designed for static data sets. As such, they cannot be applied to streaming data which are continuous, transient, and usually unbounded. Moreover, in streaming applications, there is a need to offer strong guarantees on the maximum allowed delay between incoming data and the corresponding anonymized output. Finally, technology of continuous query optimization over data streams was investigated. Monitoring aggregates on network traffic streams is a compelling application of data stream management systems. Continuously Anonymizing Streaming data via adaptive cLustEring (CASTLE), a cluster-based scheme that anonymizes data streams on-the-fly and, at the same time, the basis of the optimization is a powerful but decidable theory in which constraints over data streams can be formulated. CASTLE is extended to handle 'l-diversity'. There is a need to secure the data when transmitting it over the network. This can be done using selective encryption algorithm to compress and encrypt the data streams before transmission.

Keywords:
privacy-preserving data mining, continuous anonymity, selective security encryption

1. INTRODUCTION

Data stream Network traffic monitoring is a compelling application of data stream. Our solution is applicable in other situations where properties of a single (conceptual) entity are monitored on multiple data streams at different time instants. DATA streams are common to many application environments, such as, telecommunication, market-basket analysis, network monitoring, and sensor networks For instance, The optimization framework presented in this paper generalizes those of Kompella et al. [20] and Wang et al. [29] Recent technological advances have pushed the emergence of a new class of data-intensive applications that require continuous processing over sequences of transient data, called data streams, in near real-time. Traditional database systems have proven to be well-suited to the organization, storage, and retrieval of finite datasets. In recent years, however, new data-intensive applications have emerged that need to process data which is continuously arriving at the system in the form of potentially unbounded, time-varying sequences of data items, termed data streams involving continuous monitoring over data streams.

Data streams may contain much private information that must be carefully protected. Consider Amazon.com. In a single day, it records hundreds of thousands of online sales transactions, which are received in the form of streaming data. Suppose that the sales transaction stream has the schema S. Suppose that a relation C containing the information about Amazon customers is stored on disk, with schema. To protect customers' privacy, attributes that explicitly identify such as name, address, and telephone) are projected out of SC where mining is on SC. Mining, which needs customer information, requires joining the data stream with local customer databases. However, the remaining data in SC may still be vulnerable to linking attacks: some attributes (e.g., sex, age, and zipcode) can be exploited to re-identify individuals.

The problem is that content adaptation proxies are generally incompatible with the notion of end-to-end security. The only generic solution to this problem is the concept of selective security. The idea is to apply security selectively only to the sensitive elements of a data stream and expose the rest to any intermediary system for potential content adaptation.

None of the currently used security protocols provides an API for fine-grained control of the application of security mechanisms to a data stream because they are designed for static data sets. First, these techniques typically assume that each record in a data set is associated with a different person, that is, that each person appears in the data set only once. We propose a simple extension to the transport layer security protocol (TLS), which provides the application with an interface for selectively protecting elements within a data stream. Although this assumption is fine in a static setting, this is not realistic for streaming data. Second, data streams have a temporal dimension, since they arrive at a

Manuscript received September 5, 2013
Manuscript revised September 20, 2013

certain rate, they are dynamically processed, and the result is output with a certain delay. In some applications, the output data are immediately used to trigger appropriate procedures.
We also discuss a generic application scenario that shows how the proposed extended features can be used in conjunction with content adaptation proxies. Therefore, the application receiving the output stream should have strong guarantees on the maximum delay of the output data. (CASTLE), a cluster-based scheme k-anonymizes streams on-the-fly and, at the same time, ensures the freshness (i.e., the maximum delay between the arrival of a tuple and its release to the third party) of anonymized data by satisfying specified delay constraints. It could be the case that an expiring tuple does not belong to a cluster with size at least k. To manage this case, CASTLE implements a merge and split technique to obtain a cluster with size at least k and whose generalization minimizes the information loss.
Additionally, to reduce information loss, CASTLE exploits a strategy that allows the reuse of clusters. When a cluster is anonymized and all its tuples have been given in output, CASTLE still keeps it in memory to anonymize newly arriving tuples, if necessary.
We further discuss on methods to optimize the data streams so as to reduce the delay in transfer of the data streams.

2. THE CASTLE FRAMEWORK

CASTLE (Continuously Anonymizing STreaming data via adaptive cLustEring) is a cluster based scheme that anonymizes data streams on the fly. CASTLE groups' incoming tuples into clusters and releases all tuples belonging to the same cluster with the same generalization. CASTLE supports the anonymization of both numerical and categorical attributes, by generalizing the latter through domain generalization hierarchies, and the first through intervals.
Clustering of tuples is further constrained by the need to have fresh anonymized data. A delay constraint is added to ensure that the data transferred over the network is fresh and has not lost its meaning due to the delay between the data input and its output. This is achieved by releases the data as soon as the delay is just below the constraint acceptable i.e when a tuple is going to expire, CASTLE immediately releases it.
Clustering of data makes sure that the network is used optimally wherein a group of similar data (clusters) are transferred over the network. CASTLE defines a size of the cluster that needs to be transferred over network as k by merging or splitting the clustering when the data is about to expire making sure that there is no information loss in the process. Clusters are reused to reduce information loss so that newly arriving data streams are added to clusters after releasing the current data in the cluster.

ks-Anonymized Cluster: Let C(r1; . . . ; rn) be a cluster, and (g1; . . . ; gn) be the corresponding generalization. If at a given time instant i, size of the cluster is greater than or equal to k and all tuples in C are output with C's generalization (g1; . . . ; gn), we say that, starting from i, C is a ks-anonymized cluster.

CASTLE Algorithm

The main algorithm of Castle is Algorithm 1, which continuously processes the incoming data stream by producing in output a flow of ks-anonymized tuples.

Algorithm 1: *CASTLE*(S, k, δ, β)

```
Let $\Gamma$ be the set of non-$k_s$-anonymized clusters, initialized to be empty;
Let $\Omega$ be the set of $k_s$-anonymized clusters, initialized to be empty;
Let $\tau$ be initialized to 0;
while $S$ is non-empty do
    Let $t$ be the next tuple from S;
    Let $C$ be the cluster returned by $best\_selection(t)$;
    if $C = NULL$ then
        Create a new cluster on $t$ and insert it into $\Gamma$;
    else
        Push $t$ to $C$;
    Let $t'$ be the tuple with position equal to $t.p - \delta$;
    if $t'$ has not yet been output then
        $delay\_constraint(t')$;
```

Initially, no clusters are in memory. When CASTLE receives the first tuple, it generates a cluster over it. Then, for every newly arriving tuple t, CASTLE selects, among all the existing clusters, the one to which t can be assigned, that is, the one whose range intervals enclose t's attribute values.
However, it could be the case that no clusters can contain the new tuple, that is, the values of quasi-identifier attributes of t are not contained into the range intervals of any cluster. When a new tuple cannot be assigned to any existing cluster, there is the need to enlarge one of them in order to accommodate the new tuple. Cluster enlargement implies the enlargement of its range intervals and, as a consequence, an increase of the information loss. To minimize information loss, when selecting the cluster where a new tuple is pushed, CASTLE chooses the one that requires the smallest enlargement.

Cluster best selection

CASTLE uses the reuse strategy to prevent information loss. According to the reuse strategy, a new tuple is always pushed into a non k_s-anonymized cluster, by selecting one among those which require the minimum enlargement. Thus, to find out the best non-ks-anonymized cluster where inserting the new tuple t, best selection of cluster calculates the enlargement implied by the insertion of t in each cluster.

```
Function best_selection(t)
Let E be a set initialized empty;
foreach $C_j \in \Gamma$ do
    Let $e$ be Enlargement($C_j$, $t$);
    Insert $e$ into E;
Let $min$ be the minimum element in E;
Let $SetC_{min}$ be the set of clusters $\widetilde{C}$ in $\Gamma$ with Enlargement($\widetilde{C}$, $t$) = $min$;
foreach $C_j \in SetC_{min}$ do
    Let $\overline{IL}_{C_j}$ be the information loss of $C_j$ after pushing $t$ into it;
    if $\overline{IL}_{C_j} \leq \tau$ then
        Insert $C_j$ into $SetC_{ok}$;
if $SetC_{ok}$ is empty then
    if $|\Gamma| \geq \beta$ then
        Return any cluster in $SetC_{min}$ with minimum size;
    else
        Return NULL;
else
    Return any cluster in $SetC_{ok}$ with minimum size;
```

Tuple Output

When a tuple t is expiring, Algorithm 1 calls procedure delay constraint procedure, which checks if the tuple has been in the cluster for a long time and is approaching the expiry period in which case its main goal is to output t.

```
Procedure delay_constraint(t)
Let $C$ be the non-$k_s$-anonymized cluster to which $t$ belongs;
if $C.size \geq k$ then
    output_cluster($C$);
else
    Let $KC_{set}$ be the $k_s$-anonymized clusters in $\Omega$ containing $t$;
    if $KC_{set}$ is not empty then
        Let $\overline{KC}$ be a cluster randomly selected from $KC_{set}$;
        Output $t$ with the generalization of $\overline{KC}$;
        Return;
    Let $m$ be an integer set to 0;
    foreach $C_j \in \Gamma$ do
        if $C.size < C_j.size$ then
            $m = m + 1$;
    if $m > \frac{|\Gamma|}{2}$ then
        Suppress tuple t;
        Return;
    if $\sum_{C_i \in \Gamma} C_i.size < k$ then
        Suppress t;
        Return;
    $MC = merge\_clusters(C, \Gamma \setminus C)$;
    output_cluster($MC$);
```

It also verifies whether a merge among the cluster currently having the tuple and some of the other non-ks-anonymized clusters is possible so as to make sure that ks-anonymity is possible.

l-diversity

This principle is added to CASTLE to avoid possible inference attacks on k-anonymized data. Ensuring l-diversity requires that all tuples with the same generalization, i.e., all tuples belonging to the same group, have at least l distinct values for the sensitive attribute.
Indeed, the l-diversity principle requires to take into account during subclusters generation also the sensitive attribute, in that subclusters must have size and diversity at least equal to k and l, respectively

3. SECURITY AND OPTIMIZATION FOR CONTINUOUS DATA STREAMS

Today data streams lack security functionality. Privacy and security in the context of the streaming systems largely have been overlooked. However this is a very important topic which needs to be considered when your data is open on the web. The need to protect data and the identity of people on net has been an ever growing requirement and one of the most important concerns. This is more higher in case of data streams due to the fact that stream environments tend to be highly dynamic. Data is continuously generated and may have different security sensitivities depending on the context on personal preferences or on the stream content, all of which may frequently change.
The users owning such devices should have the ability to control their exposure to the rest of the world. Security is an important emerging requirement in software development. Beyond the potential for severe brand damage, potential financial loss and privacy issues, risk-aware customers such as financial institutions and governmental organizations are looking for ways to assess the security posture of products they build or chase. In addition, these customers plan to ultimately hold vendors accountable for security problems in their software.
A traditional approach for content access control called fully layered is to first encode data with a standard compressor and then to perform full encryption of the compressed bitstream. This scheme is relevant when the transmission of the content is unconstrained. Also this method requires altering original bitstream syntax. Selective encryption is a new trend in content protection. It consists of encrypting only a subset of the data. The main goal of selective encryption is to reduce the amount of data to encrypt while achieving a required level of security. An additional feature of selective encryption is to preserve some functionalities of the original bitstream (e.g., scalability). The general approach is to separate the content into two parts. The first part is the public part, it is left unencrypted and made accessible to all users. The second part is the protected part; it is encrypted. Only authorized users have access to protected part. One important feature in selective encryption is to make the protected part as small as possible.
Good compression is a good help for the security of selective encryption. However, most existing compression algorithms are not perfect and concentrate information energy unevenly in the bitstream. The cryptographic security should rely on the encryption key (of a well-scrutinized encryption algorithm), and unpredictability of the encrypted part.

SELECTIVE ENCRYPTION CLASSIFICATION

Selective encryption algorithm can be classified as either precompression, incompression or postcompression. Precompression algorithm performs encryption before compression and decompression before decryption. This in most cases, causes bandwith expansion which adversely impact compression efficiency and so is generally not compression friendly.

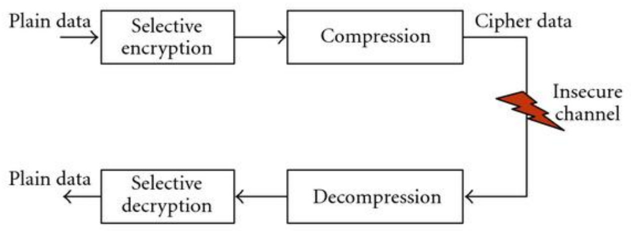


Fig: PreCompression Approach

Incompression algorithm performs joint compression and encryption which may adversely impact format compliance and compression friendliness.

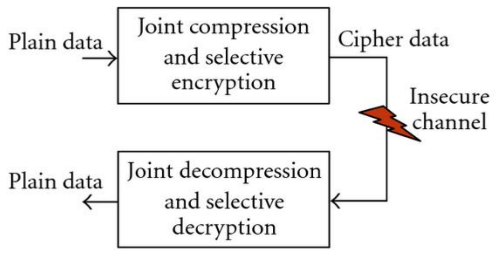

Plain data
Joint compression and selective encryption
Cipher data
Insecure channel
Plain data
Joint decompression and selective decryption

Fig: InCompression Approach

Postcompression algorithm performs compression before encryption and decryption before decompression. This algorithm is compression friendly and is inherently non-format compliant.

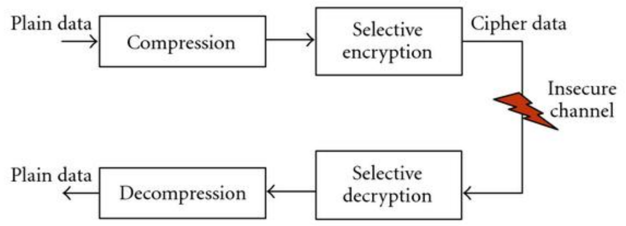


Fig: PostCompression Approach

CRITERIA FOR SELECTIVE SECURITY

There are multiple selective algorithms available and there are various criteria that help in selecting the best algorithms. The most important criteria that are applicable for data are tunability, cryptographic security and error tolerance.

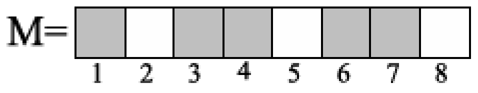
Tunability is a property for content protection systems for variant applications having different requirements and capabilities. A algorithm with dynamic encryption parameters is an ideal choice for most of the application. Information to signal the location of encrypted parts and encryption primitives and functionalities are used.

Cryptanalysis of selective encryption algorithms rely on key recovery (if encryption key space is not large enough) or prediction of encrypted part. Security of the selective encryption algorithm depends on how much and which parts of a message we have to encrypt to ensure that brute force on the encryption key space is easier than brute force attack on the plaintext itself. Otherwise, the attacker could bypass encryption and concentrate his effort on predicting the plaintext. It is hard to find an absolute measure for security. Instead, we define indirect measures that could approximate the security of a selective encryption algorithm. Perfect compression implies that all source redundancies are eliminated and that all symbols in the compressed message are independent and identically distributed.

A main challenge in selective encryption algorithms is to design secure schemes that are error tolerant. A single bit error in the encrypted part will result in many erroneous bytes in the decrypted part. As a consequence, it is important to trade off security and error tolerance.

ENCRYPTION LEVEL

Below figure shows a Message with selective encryption units with five encryption units and three unencrypted units.



The fraction of encrypted units in a message naturally defines the encryption level, how much of the message is encrypted, as follows:

Let M be a selectively encrypted message consisting of n equally sized units. Then the encryption level, EL, of M is defined as the ratio EL = ne/n where ne is the number of encrypted units in M.

The encryption and compression of data streams can be done prior to applying the CASTLE Algorithm to convert data streams into ks-anonymized clusters. The selective security makes sure that the data is compressed thereby optimizing the transfer of data since the data size is reduced and thereby reducing the network bandwidth. The selective security optimizes the transfer of data streams over the network and also secures the data thereby maintaining the privacy of the data and the individuals.

4. CONCLUSIONS:

In this paper, CASTLE a cluster-based framework for continuous data streams is optimized and secured by using selective security mechanism. Relevant features of CASTLE are the enforcement of delay constraints, its adaptability to data distributions, and its cluster reuse strategy that improves the performance without compromising security. This algorithm is extended by applying security and compressing data before passing the data over network. There have been a number of efforts focusing on distributed processing of stream queries in general and optimization in particular.

ACKNOWLEDGEMENTS:

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowned our efforts with success. We extend our deep sense of gratitude to Principal Dr. S. M. Afroz, and HOD, Computer Science and Engineering, Nizam Institute of Engineering and Technology, Deshmukhi, Mr.M.S.Qaseem, HOD, Information technology, Nizam Institute of Engineering and Technology, Deshmukhi, for his support and encouragement. I am indebted to Ms. Asma, Associate professor, Nizam Institute of Engineering and Technology, Deshmukhi, Nalgonda (A.P), India for giving her helpful comments and sharing ideas in carrying out this research work.

REFERENCES:

[1] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu, "A Framework for Clustering Evolving Data Streams," Proc. Int'l Conf. Very Large Databases (VLDB), pp. 81-92, 2003.

[2] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D.Thomas, and A. Zhu, "Achieving Anonymity via Clustering,"Proc. Symp. Principles of Database Systems (PODS), pp. 153-162,2006.

[3] J. Cao, B. Carminati, E. Ferrari, and K.L. Tan, "CASTLE: A Delay- Constrained Scheme for k s-Anonymizing Data Streams," Proc. Int'l Conf. Data Eng. (ICDE), Poster Paper, pp. 1376-1378, 2008.

[4] CASTLE: Continuously Anonymizing Data Streams, May/June 2011. Jianneng Cao, Barbara Carminati, Elena Ferrari, Kian-Lee Tan CE Shannon, Communication theory of secrecy systems Declassified Report, 1946

[5] Overview on Selective Encryption of Image and Video: Challenges and Perspectives. "A Massoudi*, F Lefebvre, C De Vleeschouwer, B Macq and J-J Quisquater

S. Nasira Tabassum received her Master of Computer Application from Muffakham Jah College of Engineering and Technology, Affiliated to Osmania University, Hyderabad, AP India. She is currently pursuing M.Tech in Software Engineering from Nizam Institute of Engineering and Technology, Deshmukhi, Nalgonda Dist, Affiliated to JNTU Hyderabad, AP India. Her areas of interest include web technologies, data warehousing techniques, Artificial Intelligence, Image Processing, network security and Data Mining