International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-2 Issue-1, October 2012

Optimization and Security of Continuous Anonymizing Data Streams

S. Nasira Tabassum

Abstract*— 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. Continuously Anonymizing Streaming data via adaptive cLustEring (CASTLE), an efficient and effective algorithm w.r.t. the quality of the data, is a cluster-based scheme that anonymizes data streams on-the-fly and, at the same time, ensures the freshness of the data. CASTLE is also extended to handle l-diversity. Finally, we study the optimization and security techniques of data streams using selective security encryption and compression to improve the efficiency of the CASTLE algorithm.

Index Terms*— privacy-preserving data mining, continuous anonymity, selective security encryption, data compression.

I. 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. Mining these continuous data streams [1] helps companies to learn the behavior of their customers or entities thus providing means to analyze the data which is useful in bringing unique opportunities. DATA streams are common to many application environments, such as, telecommunication, market-basket analysis, network monitoring, and sensor networks. 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.

Manuscript published on 30 October 2012.
***** Correspondence Author (s)
S. Nasira Tabassum, M.Tech (Department of SE), Nizam Institute of Engineering and Technology, Deshmukhi, Nalgonda, Andhra Pradesh, India.

Since most companies do not have in-house expertise of data mining, they have to outsource the mining to professional third party companies. Data streams may contain much private information that must be carefully protected [6] or carefully anonymized [4] before they are passed to the third party. Consider an online sales application where in a single day, it may record hundreds of thousands of online sales transactions. This information is 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 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 [9]. 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. 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.

A data set satisfies k-anonymity [3], [12] property if each combination of values in the data set occurs at least k times. A well-known technique to achieve k-anonymity is generalization where an attribute is replaced by a less specific but semantically related value. However, traditional k-anonymity schemes are not suitable for streaming data. The main reason is that these methods cannot be directly applied on streaming data, 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. 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 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. k-anonymity of data streams is denoted by ks-anonymity to distinguish it from k-anonymity for static data sets. A relevant property of ks-anonymized data is their freshness which is considered as the maximum allowed time of a tuple staying in the memory before it is output.

We will discuss a generic application scenario that shows how the proposed extended features of CASTLE can be used in conjunction with content adaptation proxies of selective encryption. CASTLE algorithm is used to anonymize the data streams and group the data into clusters. These clusters are then transferred over the network. This ensures freshness of the data by satisfying specified delay constraints. We discuss on methods to optimize and secure the data streams to reduce the delay in transfer of the data streams and secure the data streams thus preserving privacy of the customers. Selective encryption algorithm is used to encrypt the data to ensure security and optimize transfer by compressing the data.

II. THE CASTLE FRAMEWORK

CASTLE (Continuously Anonymizing STreaming data via adaptive cLustEring) [5], [7] 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 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. CASTLE groups incoming tuples into clusters, each having size at least k to achieve k-anonymity. All these tuples belonging to the same cluster with the same generalization are then released. 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. CASTLE is also extended to support l-diversity [10] on data streams.

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.

CASTLE supports the anonymization of both numerical and categorical attributes, by generalizing the latter through domain generalization hierarchies, and the first through intervals.

Each tuple received is assigned to a cluster having the range intervals within the attribute values. 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.

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 releasing 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) is transferred over the network. CASTLE defines a size of the cluster that needs to be transferred over network 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.

III. 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)

```
1  Let Γ be the set of non-k_δ-anonymized clusters, initialized to be empty;
2  Let Ω be the set of k_δ-anonymized clusters, initialized to be empty;
3  Let τ be initialized to 0;
4  while S is non-empty do
5      Let t be the next tuple from S;
6      Let C be the cluster returned by best_selection(t);
7      if C = NULL then
8          Create a new cluster on t and insert it into Γ;
9      else
10         Push t to C;
11     Let t' be the tuple with position equal to t.p − δ;
12     if t' has not yet been output then
13         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 ks-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.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Function *best_selection*(t)

1 Let E be a set initialized empty;
2 **foreach** $C_j \in \Gamma$ **do**
3 Let e be Enlargement(C_j, t);
4 Insert e into E;
5 Let min be the minimum element in E;
6 Let $SetC_{min}$ be the set of clusters $\widetilde{C}$ in Γ with Enlargement($\widetilde{C}$, t) = min;
7 **foreach** $C_j \in SetC_{min}$ **do**
8 Let $\overline{IL}_{C_j}$ be the information loss of C_j after pushing t into it;
9 **if** $\overline{IL}_{C_j} \leq \tau$ **then**
10 Insert C_j into $SetC_{ok}$;
11 **if** $SetC_{ok}$ *is empty* **then**
12 **if** $|\Gamma| \geq \beta$ **then**
13 Return any cluster in $SetC_{min}$ with minimum size;
14 **else**
15 Return NULL;
16 **else**
17 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)

1 Let C be the *non-k_s*-anonymized cluster to which t belongs;
2 **if** $C.size \geq k$ **then**
3 output_cluster(C);
4 **else**
5 Let KC_{set} be the k_s-anonymized clusters in Ω containing t;
6 **if** KC_{set} *is not empty* **then**
7 Let $\overline{KC}$ be a cluster randomly selected from KC_{set};
8 Output t with the generalization of $\overline{KC}$;
9 Return;
10 Let m be an integer set to 0;
11 **foreach** $C_j \in \Gamma$ **do**
12 **if** $C.size < C_j.size$ **then**
13 $m = m + 1$;
14 **if** $m > \frac{|\Gamma|}{2}$ **then**
15 Suppress tuple t;
16 Return;
17 **if** $\sum_{C_i \in \Gamma} C_i.size < k$ **then**
18 Suppress t;
19 Return;
20 MC =*merge_clusters*$(C, \Gamma \setminus C)$;
21 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.

When outputting a cluster C, in order to minimize information loss, procedure output_cluster() verifies if the cluster can be split, i.e., whether its size is at least 2k. If this is the case, it calls function that splits C into subclusters, each with size at least k. Then, all tuples of the newly created clusters (or of C, respectively) are given in output with the generalization of the corresponding cluster.

Procedure *output_cluster*(C)

1 **if** $C.size \geq 2k$ **then**
2 Let SC be the set of clusters returned by $split(C)$;
3 **else**
4 $SC = \{C\}$;
5 **foreach** $C_i \in SC$ **do**
6 Output all tuples in C_i with its generalization;
7 Update τ according to *InfoLoss*(C_i);
8 **if** InfoLoss(C_i) $< \tau$ **then**
9 Insert C_i into Ω;
10 **else**
11 delete C_i;
12 delete C_i from Γ;

The average information loss r of the most recent ks-anonymized clusters including the new ones is updated to InfoLoss procedure. Output_cluster() function does not store all new clusters into set of ks-anonymized clusters denoted by **Ω**. To minimize information loss, only clusters with good information quality are reused. For this reason, output cluster() inserts a new ks-anonymized cluster Ci into **Ω** only if its information loss is less than r (steps 8-11). In addition, Ci is deleted from the set of non ks anonymized clusters (step 12).

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

IV. 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 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.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

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. In traditional content protection schemes, called fully layered, the whole content is first compressed. Then, the compressed bitstream is entirely encrypted using a standard cipher (DES, AES, IDEA, etc.). The specific characteristics of this kind of data (high-transmission rate with limited bandwidth) make standard encryption algorithms inadequate. Another limitation of fully layered systems consists of altering the whole bitstream syntax which may disable some codec functionalities. This scheme is relevant when the transmission of the content is unconstrained. Also this method requires altering original bitstream syntax. The total encryption and decryption process are computationally demanding and time consuming, thus not suitable for real time communication.

Some recent works explored a new way of securing the content, named, partial encryption or selective encryption, soft encryption, perceptual encryption, by applying encryption to a subset of a bitstream. Selective encryption [10], [15] is a new trend in content protection. Only a subset of the entire data is encrypted instead of the complete data stream. Encryption of only a selected portion of the data stream lowers the computational load both at the user as well as server end. Selective encryption strives for computational complexity rather than for maximum security. It is best employed in the transform domain as it is easier to identify what parts of data are critical for security, allowing different levels of security and transparency. Moreover, it is easier to locate the selected data in frequency domain without any processing overhead.

The main goal of selective encryption [14] 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.

V. SELECTIVE ENCRYPTION CLASSIFICATION

Selective encryption algorithm can be classified as precompression, incompression or postcompression.

A. Precompression algorithm

Precompression algorithm performs encryption before compression and decompression before decryption. This in most cases, causes bandwidth expansion which adversely impact compression efficiency and so is generally not compression friendly.

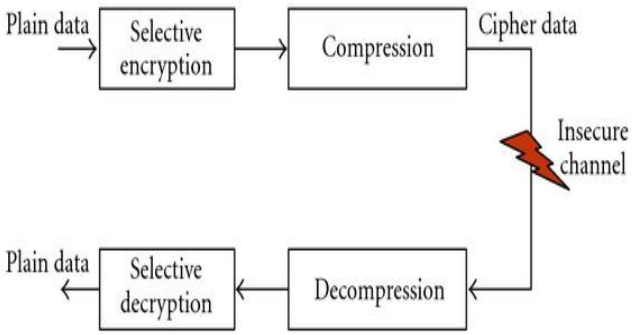

Fig 1: PreCompression Approach

B. Incompression algorithm

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

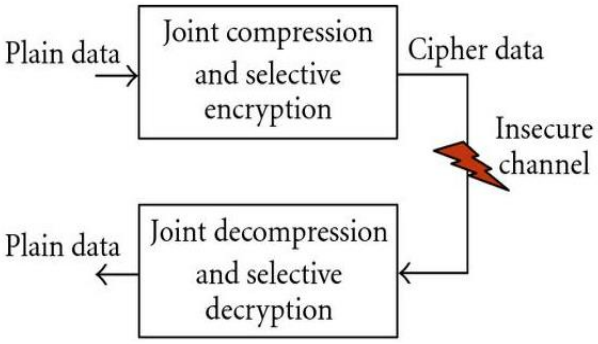

Fig 2: InCompression Approach

C. Postcompression algorithm

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

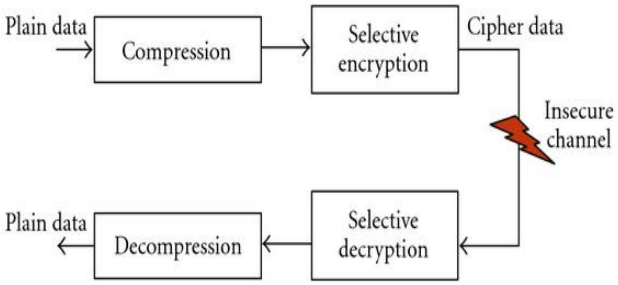

Fig 3: PostCompression Approach

VI. 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.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

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.

VII. 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 selective security can be added to the castle algorithm to optimize the transfer of data streams over the network and also to secure the data thereby maintaining the privacy of the data and the individuals.

The encryption and compression logic of selective encryption algorithm can be merged with the CASTLE Algorithm to apply compression and encryption steps before the streams are divided into ks-anonymized clusters. The selective security algorithm compresses the data thereby reducing the network bandwidth due to decrease in data size. Since the data is encrypted before transferring it over the network, privacy of the data and the individuals is achieved. This helps in reducing the linking attacks that are used to re-identify individuals by attackers.

VIII. CONCLUSION

In this paper, we have presented CASTLE a cluster-based framework to k-optimize data streams. Relevant features of CASTLE are the enforcement of delay constraints, its adaptability to data distributions, cluster enlargement and split to ks-anonymize data streams and its cluster reuse strategy that improves the performance without compromising security. We have extended this algorithm by applying security and compressing data before passing the data over network by using selective encryption mechanism. This mechanism helps in optimizing the data transfer over the network in addition to securing the data against potential attacks by unauthorized users.

IX. ACKNOWLEDGMENT

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] P. Domingos and G. Hulten, “Mining High-Speed Data Streams,” Proc. Int’l Conf. Knowledge Discovery and Data Mining (KDD), pp. 71-80, 2000.
[2] 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.
[3] C.C. Aggarwal, “On k-Anonymity and the Curse of Dimensionality,” Proc. Int’l Conf. Very Large Databases (VLDB), pp. 901-909, 2005.
[4] 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.
[5] 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.
[6] V.S. Iyengar, “Transforming Data to Satisfy Privacy Constraints,” Proc. Int’l Conf. Knowledge Discovery and Data Mining (KDD), pp. 279-288, 2002.
[7] Jianneng Cao, Barbara Carminati, Elena Ferrari, Kian-Lee Tan “CASTLE: Continuously Anonymizing Data Streams”, May/June 2011.
[8] CE Shannon, Communication theory of secrecy systems Declassified Report, 1946
[9] Marius Portmann, Aruna Seneviratne, "Selective Security for TLS," icon, pp.216, Ninth IEEE International Conference on Networks (ICON'01), 2001
[10] A Massoudi*, F Lefebvre, C De Vleeschouwer, B Macq and J-J Quisquater “Overview on Selective Encryption of Image and Video: Challenges and Perspectives. “EURASIP Journal on Information Security 2008, 2008:179290
[11] A. Machana vajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, “L-Diversity: Privacy beyond k-Anonymity,” Proc. Int’l Conf. Data Eng. (ICDE), p. 24, 2006.
[12] R.J. Bayardo and R. Agrawal, “Data Privacy through Optimal k-Anonymization,” Proc. Int’l Conf. Data Eng. (ICDE), pp. 217-228, 2005.
[13] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, “Fast Data Anonymization with Low Information Loss,” Proc. Int’l Conf. Very Large Databases (VLDB), pp. 758-769, 2007.
[14] Tom Lookabaugh, Douglas C. Sicker, University of Colorado at Boulder, “Selective Encryption for Consumer Applications”.
[15] Nidhi S Kulkarni, Balasubramanian Raman, and Indra Gupta, “Selective Encryption of Multimedia Images”, XXXII NATIONAL SYSTEMS CONFERENCE, NSC 2008, December 17-19, 2008
[16] Hwayoung Um and Edward J. Delp, "Selective Video Encryption Of A Distributed Coded Bitstream Using LDPC Codes", SPIE-IS&T/ Vol. 6072 60721B-1
[17] Pu Wang, Jianjiang Lu, Lei Zhao, Jiwen Yang, "B-CASTLE: An Efficient Publishing Algorithm for K-Anonymizing Data Streams," gcis, vol. 2, pp.132-136, 2010 Second WRI Global Congress on Intelligent Systems, 2010
[18] W. Puech, A. G. Bors, J. M. Rodrigues, "Protection of Colour Images by Selective Encryption", Advanced Color Image Processing and Analysis, 2013, pp 397-421

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

[19] Tom Lookabaugh*, Douglas C. Sicker, David M. Keaton, Wang Ye Guo, Indrani Vedula, "Security Analysis of Selectively Encrypted MPEG-2 Streams"
[20] Shaimaa A. El-said, Khalid F. A. Hussein, Mohamed M. Fouad, "Securing Image Transmission Using In- Compression Encryption Technique", International Journal of Computer Science and Security, (IJCSS), Volume (4): Issue (5)
[21] Marc Van Droogenbroeck and Raphaël Benedett, "Techniques for a selective encryption of uncompressed and compressed images", Advanced Concepts for Intelligent Vision Systems (ACIVS), Ghent, Belgium, pages 90-97, September 2002
[22] A. Massoudi, F. Lefebvre, C. De Vleeschouwer, B. Macq, and J.-J. Quisquater, "Overview on Selective Encryption of Image and Video: Challenges and Perspectives", EURASIP Journal on Information Security Volume 2008, Article ID 179290

S. Nasira Tabassum has received her Master's in Computer Application from Muffakham Jah College of Engineering and Technology, Affiliated to Osmania University, Hyderabad, AP India. She is currently pursuing Master's in Technology (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 and Data Mining.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)