**USING OUTSOURCED ENCRYPTED MEDICAL IMAGES, A QUICK NEAREST NEIGHBOUR SEARCH SCHEME**

**ABSTRACT**:

Medical imaging is crucial for medical diagnosis, and the sensitive nature of medical images necessitates rigorous security and privacy solutions to be in place. In a cloud-based medical system for Healthcare Industry 4.0, medical images should be encrypted prior to being outsourced. However, processing queries over encrypted data without first executing the decryption operation is challenging and impractical at present. In the paper, we propose a secure and efficient scheme to find the exact nearest neighbor over encrypted medical images. Instead of calculating the Euclidean distance, we reject candidates by computing the lower bound of Euclidean distance that is related to the mean and standard deviation of data. Unlike most existing schemes, our scheme can obtain the exact nearest neighbor rather than an approximate result. We then evaluate our proposed approach to demonstrate its utility.

|  |  |
| --- | --- |
| **EXSISTING SYSTEM** | **PROPOSED SYSTEM** |
| * There exists a body of literature studying auction-based resource allocation in other contexts such as spectrum sharing in cognitive networks. * Existing frameworks only consider one type of cloud servers and one type of tasks that cannot capture the reality of the market. | * In this paper, we investigate both interactions through a two-stage auction mechanism. For the interactions between customers and cloud managers, we adopt the options-based sequential auctions (OBSAs) to design the cloud resource allocation paradigm. * As compared to existing works, our framework can handle customers with heterogeneous demands, provide truthfulness as the dominant strategy, enjoy a simple winner determination procedure, and preclude the delayed entrance issue. We also provide the performance analysis of the OBSAs, which is among the first in literature. |
| **EXISTING ALGORITHM**  Sequential Auction | **PROPOSED ALGORITHM: -**  Options-Based Sequential Auctions (OBSAs) |
| **ALGORITHM DEFINITION: -**  In sequential auction, the seller holds consecutive auctions for selling goods. These are suitable for the following scenarios: (i) availability of the goods varies over time, which means the goods may not be available in some of the time instances; (ii) the buyers arrive at the market at different times, which requires the seller to wait for some period of time before the number of buyers exceeds a threshold to guarantee a certain profit. Nevertheless, one of the main drawbacks of these auctions is the lack of a dominant strategy that can accommodate heterogeneous demands of buyers when customers face multiple sequential auctions. | **ALGORITHM DEFINITION: -**  Our options-based sequential auctions (OBSAs) is used to design the cloud resource allocation paradigm. As compared to existing works, our framework can handle customers with heterogeneous demands, provide truthfulness as the dominant strategy, enjoy a simple winner determination procedure, and preclude the delayed entrance issue. The existence of the price matching process makes the analysis of OBSAs completely different from that of classic sequential auctions. In OBSAs we model the price matching process of an observer. One of the important performance metrics of any auction is the auctioneer’s income or profit which will be considered in our proposed mechanism. |
| **DRAWBACKS: -**   * Incapability of handling customers’ heterogeneous demands that require a bundle of different types of servers. * Missing the truth-fullness property. * Requiring prohibitive computation for winner and payment determination. * Susceptibility to the delayed entrance issue. | **ADVANTAGES: -**   * Uniform distribution * Better price matching process * Proving transparency |

**MINIMUMSYSTEM REQUIREMENTS**

**HARDWARE REQUIREMENTS**:

System : Pentium i3 Processor

Hard Disk : 500 GB.

Monitor : 15’’ LED

Input Devices : Keyboard, Mouse

RAM : 2 GB

**SOFTWARE REQUIREMENTS:**

Operating system : Windows 10.

Coding Language : Java

Tool : Eclipse

Database : MYSQL