Research Paper

GYAN MANAGEMENT
AN INTERNATIONAL BI-ANNUAL REFEREED JOURNAL OF MANAGEMENT & TECHNOLOGY
Year 2024, Volume-18, Issue-2 (July - December)
ISSN 0974-7621 (Print)

Human resource transaction management system security optimize using multithreading with blowfish Algorithm

Ashu Krishna[1], Satyajit Rath[2] & Madhurendra Kumar[3]

[1]Research Scholar, Mewar University, Chittorgarh, Rajasthan 312901, India
[2]CNeM, CSIR-IMMT, Bhubaneswar 751013, India 312901, India
[3]CDAC, Noida 201307 UP, India

ARTICLE INFO

Key Words: *HRMS, Blowfish algorithm,* Data, *Multithreading, and management*

ABSTRACT

Today HRMS (Human resource management system) application around data transmitting from one place to another is challenging and doing this with real-time data is even more difficult to service time minimize and secure. In this paper using Blowfish algorithm with multithreading technology we will solve the waiting time using distributed data streaming platform. This approach will become ideal to daily life where each and every user get services in optimize way with secure direction.

Introduction

Today Human Resource [HR] application around data transmitting from one place to another is challenging and doing this with real-time data is even more difficult to service optimize and secure under Health ERP. In this paper using Blow fish algorithm with multithreading technology we will solve the waiting time using distributed data streaming platform. This approach will be become ideal to daily life where each and every user get services in optimize way with secure direction. It is critical for companies to align their human resources to better achieve their strategic goals. If you don't the duration, assets along with vitality will be wasted. An organization's alignment with its corporate goals can be enhanced by reviewing hiring procedures, communicating mission and vision statements, using shared goals, designing effective reward systems, empowering staff, and promoting team development within the organization. HR administration is the management function that helps leaders to plot, pay, recruit, appoint, supervise, grow, pay off and retain the executive members. HR management pursues four objectives: social, organizational, functional and

Received 15.12.2023; Accepted 02.02.2024
DOI: 10.48165/gmj.2023.18.2.4

personal development. Organizations must develop policies. It has clear procedures and clear policies for its people, which contribute to the effectiveness, continuity and stability of the organization. HRMS communication utilizing blowfish and multithreading with open space in an optimize the secure communication The optimal secure communication between substances requires a secure method that prevents third parties from spying on them, ensuring they communicate without interference or listening in. Secure communication refers to the methods applied for inurement the truthfulness and its privacy of data transposed between individuals, in such a way that third parties cannot captured what they say. Beyond face-to-face discussions that cannot be listened stealthily, laws, assets, specialized issues (capture attempts and encryption), and sheer volume of communications offer assistance constrain observation. Innovation and its encroachments are at the center of this talk, as many communications take put over long separations and through innovation, and there's a developing mindfulness of the significance of the issue of capture attempts. For this reason, this article centers on communications that are intervened or catching by innovation. See too Trusted Computing. Typically, an approach right now beneath advancement that gives in general security at the potential taken a toll of constrained dependence on businesses and government offices.

After execution of this concept it'll be guarantee to fast communication and diminish holding up time amid surge hours. Too it guarantees an supreme prepare without making client stand in holding up. Concurring to the back staff were posted at the clinic to assist the client get it the steps and benefits of utilizing this benefit. Where imperative part will be play utilize of great adjusting in Blow angle secure innovation and administrations is or maybe straightforward when managing with effortlessly unsurprising workloads and ideally dependable accessible server.

Blowfish algorithm

Bruce Schneier's 1993-developed Blowfish is a popular symmetric-key block cipher encryption technique known for its quick encryption and decryption processes. This post explores its major characteristics, encryption process, and Java implementation.

Centre for attraction of Blowfish

Block Cipher: Is a block cipher which encrypts each individual block of sixty-four bits' of plaintext.

- **Symmetric Key Encryption:** It encompasses the encryption method using a symmetric key to ensure the same key is used for encryption and decryption, to and fro.

Variable Key Size: It converts the encrypted block in a variable key sizes of up to four hundred forty eight bits, making it more safe than unadventurous encryption practices.

- **Feistel Cipher:** Feistel Cipher structure is used to encrypt the block data and dole out plaintext into two cut up and recursively encrypts each half using mathematical operations.

Encryption Process

The Blowfish's encryption procedure utilized below mentioned ladders:

- **Key Generation:** The key expansion algorithm is used for production of encryption key to the original key and generates a series of sub keys.
- **Initial Permutation:** The first permutation dispatches the 64-bit plaintext.
- **Splitting:** The -bit block is divided into two half, each having thirty-two bits.
- **Rounds:** Blowfish is a block game with 16 rounds, each involving a complex sequence of replacements and permutations to both blocks.
- **Final Permutation:** With completion of sequences of sixteen, a last permutation is applied to the encrypted text after a last sequence.

Multithreading

Multithreading is a computer execution method that allows several threads to be created within a process, each of which runs independently yet shares process resources. Threads, depending on the hardware, can execute in perfect parallel if distributed throughout any process.

The major reason for adding threads to a program is to increase performance. Performance can be expressed in a variety of ways:

- Web servers employ many threads to handle data requests concurrently.
- An image analysis algorithm launches multiple threads simultaneously, segmenting a picture into quadrants before applying filtering to the image.
- A ray-tracing program runs many threads to compute visual effects, with the main GUI thread rendering the final results.

Additionally, multithreading optimizes and minimizes the utilization of computer resources. Because requests from one thread do not impede requests from other threads, application responsiveness increases.

The fact that multithreading requires less resource intensity than executing many processes at once should not be overlooked. Compared to establishing and managing threads, creating progressions involves a lot more overhead, time consumption, and regulating..

Compare-And-Swap optimization

Compare-and-swap could be a method utilized in multithreading to supply non-blocking string security. So distant what we have seen with synchronized and Reentrant Bolt, are the blocking components. This CAS procedure is indeed actualized at the hardware level right into the machine's Instruction Set. For case, within the Intel x-86, it is executed as CMPXCHG (compare-and-exchange) instruction. All the advanced multiprocessor models back CAS in their instruction set. It is the foremost well known primitive for executing non-blocking concurrent collections. Most of the concurrent collections in Java utilize CAS in combination with minimal locking (Lock Striping) to realize a better degree of concurrency. To get it how CAS works, consider our Counter situation, where we have two strings T1 and T2 and both are attempting to increase the esteem of the Counter protest. We know that the increase operation isn't nuclear. It really isolates into three nuclear operations: Perused, Increase, and Type in. And it is at the Type in operation the CAS comes into the picture.

My Proposed work

My research methods based on multithreading java concept using blowfish algorithm where there is an improvement in the time involvedness and security in a group of consumer and application server. Now we'll illustrate a classic interaction between two threads: application server and a Consumer. An application server A thread generates messages and queues them, while a customer declaims and shows the output. For the sake of realism, we set the queue's maximum depth. To make things more interesting, we will also make our consumer thread considerably lazier and slower than the Communication thread. This means that algorithm at application server be able to implement reduce the duration complexity between consumer and application server. Now we'll illustrate a classic interaction between two threads: A application server and a Consumer. An application server thread generates messages and adds them to a queue, which a client may later read and see.

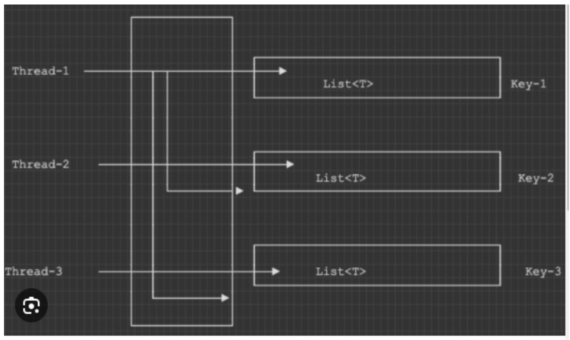

Fig. 1. Work flow diagram

Algorithm

Actual steps Blowfish algorithm using Multithread follows in order to encrypt:

```
    Require: DataSeries: series of data
    Require: EN: number of element in the series
    Ensure: DataSeries: series of processed data
EN is 64 bits input data
EN is divided into two equal parts en1 and en2
For i=0 to 15 EN1=en1 xor Pi EN2=f(en1) xor en2
    // Process optimize using Multithreading concept
    //*
    Function THREADPROCESS (lower, upper, DataSeries, EN)
    for i = lower to upper do
     for j = lower + 1 to N do
    if (DataSeries[i].P1 R A[j].P1) then .              R is the
    relation between elements
    DataSeries [i].P2 = DataSeries [i].P2 ∪ {A[j]}
    DataSeries [j].P2 = DataSeries [j].P2 ∪ {A[i]}
EN1=en2 xor P18
EN2=en2 xor P17
Combine en1 and en2
     end if
          End for
                 End for
                        Return DataSeries
                                End function
    */I
```

For decryption, this process is applied, except that the sub-keys Pi must be provided in opposite direction.

Program Implementation

```
//*
import java.io.UnsupportedEncodingException;
import java.nio.charset.Charset;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;
// Security optimize time
public class BlowfishThread {
   public void run()
   {
public String encrypt(String password, String key)
throws
       NoSuchAlgorithmException, NoSuchPadding
Exception,
        InvalidKeyException, IllegalBlockSizeException,
          BadPaddingException, UnsupportedEncoding
Exception {
     byte[] KeyData = key.getBytes();
      SecretKeySpec KS = new SecretKeySpec(KeyData,
"Blowfish");
     Cipher cipher = Cipher.getInstance("Blowfish");
     cipher.init(Cipher.ENCRYPT_MODE, KS);
     String encryptedtext = Base64.getEncoder().
                encodeToString(cipher.doFinal(password.
getBytes("UTF-8")));
     return encryptedtext;
  }
  public String decrypt(String encryptedtext, String key)
       throws NoSuchAlgorithmException, NoSuch
PaddingException,
        InvalidKeyException, IllegalBlockSizeException,
               BadPaddingException {
     byte[] KeyData = key.getBytes();
      SecretKeySpec KS = new SecretKeySpec(KeyData,
"Blowfish");
     byte[] ecryptedtexttobytes = Base64.getDecoder().
               decode(encryptedtext);
     Cipher cipher = Cipher.getInstance("Blowfish");
     cipher.init(Cipher.DECRYPT_MODE, KS);
     byte[] decrypted = cipher.doFinal
(ecryptedtexttobytes);
     String decryptedString =
     new String(decrypted, Charset.forName("UTF-8"));
     return decryptedString;
  }
     try {

     }
     catch (Exception e) {
        // Throwing an exception
        System.out.println("Exception is caught");
     }
  }
  public static void main(String[] args) throws Exception
{
BlowfishThread obj=new BlowfishThread ();
Thread  t =new Thread  (obj);
final String password = "Cdac@123";
     final String key = "cdac12345";
     System.out.println("Password: " + password);
     BlowfishDemo obj = new BlowfishDemo();
     String enc_output = obj.encrypt(password, key);
    System.out.println("Encrypted text: " + enc_output);
     String dec_output = obj.decrypt(enc_output, key);
    System.out.println("Decrypted text: " + dec_output);
     t.start();
  }
}
}
*//
```

Output:

Password: KKTr@123
Encrypted text: 4DTHqnctCuk=
Decrypted text: KKTr@123
The research' applications results show that the suggested Multithreading Blowfish algorithm has the following advantages over the current Blowfish algorithm:

- The first The benefit is the identical input is converted to encrypted text, significantly improving the ideal time security feature. This is because a different random number is generated in each round, which causes variations in the multithreading blow-fish algorithm function's performance.
- The second major benefit is that it takes less time than Blowfish algorithm because the Multithreading parallel processing is used.
- 3rd advantage is higher throughput.
- 4th advantage is high security metric.
- 5th advantage is Avalanche value.
- 6th advantage is high efficiency.
- 7th advantage is high throughput.
- 8th advantage is higher efficiency.

In nutshell Blowfish algorithm with multithreading is much more efficient compared to blowfish algorithm. The above results clearly show that blowfish algorithm with multipronged reading is more efficient than blowfish algorithm in terms of encryption time, decryption time, throughput, Avalanche effect, and Power consumption. Blowfish Multithreading algorithm can be used on consumer electronic devices like personal digital assistants (PDAs) and smart phones, which consume less memory and consume less power. It can be used on personal database programs and can be used to encrypt in removable media. It can be used for clinical data collection and biometrics such as voice, facial, or finger print authentication. This study can be extended further with optimization techniques that have high potential.

Conclusion

Conclusion the Blowfish Algorithm with multithreading was thoroughly studied. The algorithm was implemented as a software application using Java. The software application demonstrated the encryption and decryption processes successfully.

After implementation of this concept it will be ensure to quick communication and reduce waiting time during rush hours. Also it ensures an absolute process without making client stand in waiting. According to the support personnel were posted at the organization to help the HRMS User understand the steps and benefits of using this service. Where important role will be play use of good balancing in Blow fish secure technology and services is rather simple when dealing with easily predictable workloads and ideally reliable available server.

References

Dr. Madhurendra Kumar,"Cloud computing Network Problem and Storage solutions Using Ant colony optimization" International Journal of Scientific & Engineering Research Volume 7, Issue 12, December-2016, ISSN 2229-5518

Dorigo M. and Blum C., "Ant Colony Optimization Theory: A Survey," in Theoretical Computer Science, vol. 344, no. 2, pp. 243-278, 2005.

Dorigo M., Birattari M., and Stutzel T., "Ant Colony Optimization," IEEE Computational Intelligence Magazine, vol. 1, no. 4, pp. 28-39, 2006.

Fangzhe C., Ren J., and Viswanathan R., "Optimal Resource Allocation in Clouds," in Proceedings of the 3rd International Conferenceon Cloud Computing, Florida, USA, pp. 418- 425, 2010.

Gao K., Wang Q., and Xi L., "Reduct Algorithm Based Execution Times Prediction in Knowledge Discovery Cloud Computing Environment," the International Arab Journal of Information Technology, vol. 11, no. 3, pp. 268- 275, 2014.

Gao Y., Guan H., Qi Z., Hou Y., and Liu L., "A Multi-Objective Ant Colony System Algorithm for Virtual Machine Placement in Cloud Computing," Journal of Computer and System Sciences, vol. 79, no. 8, pp. 1230-1242, 2013.

R. Buyya and M. Murshed. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. Concurrency and omputation: Practice and Experience, 14(13-15), Wiley Press, Nov.-Dec., 2002.

Ghalem B., Tayeb F., and Zaoui W.,"Approaches to Improve the Resources Management in the Simulator Cloudism," in Proceedings of the Conference on Interactionand Confidence Building Measures in Asia,Lecture Notes in Computer Science, Istanbul, Turkey, pp. 189-196, 2010.

Hsu C. and Chen T., "Adaptive Scheduling Based on Quality of Service in Heterogeneous Environments," in Proceedings of the IEEEInternational Conference on Multimedia and Ubiquitous Engineering, California, USA, pp. 1-6, 2010.

Ijaz S., Munir E., Anwar W., and Nasir W.,"Efficient Scheduling Strategy for Task Graphs in Heterogeneous Computing Environment," theInternational Arab Journal of Information Technology, vol 10, no. 5, pp. 486-492, 2013.

Kessaci Y., Melab N., and Talbi E., "A Pareto- Based GA for Scheduling HPC Applications on Distributed Cloud Infrastructures," in Proceedings of the IEEE International Conference on High Performance Computing and Simulation, Istanbul, Turkey, pp. 456-462, 2011