International Journal of Future Generation Communication and Networking
Vol. 13, No. 2, (2020), pp. 439 – 464

Enforcing Security in Replicated DRTDBS

Pratik Shrivastava[1], Udai Shanker[2]

[1,2]*Department of CSE, MMMUT, Gorakhpur*

Abstract

Timeliness and temporal consistency are the two primary requirements of DRTDBS (Distributed Real-Time Database System). Timeliness is express in the form of a deadline, and the scheduler assigns the priority based on the timeline. There exist different non-deterministic factors in the DRTDBS that make it challenging to meet the deadline timely. Distributed processing, limitation of system memory, the existence of single version data item, and access latency are some of them. The replication technique is incorporated into this system to extricate such issues. This technique improves the timeliness but compromises in mutual consistency. Replication protocol is design to simultaneously improves timeliness and mutual consistency between data replicas in the system.

Existing replication protocol neither prevents the occurrence of a covert channel nor prevents the unauthorized access. Thus, embedding security in the replicated DRTDBS (RDRTDBS) becomes an essential part of the research for the researchers. In the present paper, our main objective is to prevent covert channels in between low clearance level and high clearance level real-time transactions (RTTs) and to prevent the unauthorized access of messages propagating in the network. Our proposed solution is incorporated in the system model (Shrivastava, P., Shanker, U., 2018c) such that our solution work for inside and outside security in the homogeneous working environment of RDRTDBS. We have implemented and tested our proposed solution using different parameters. The experimental result shows that our proposed solution is beneficial for real-time applications that demand security, timeliness, and mutual consistency.

Keywords: *Replication Protocol, Security, RDRTDBS, Covert Channel, Mutual Consistency*

Biographical notes: (ABS)

This paper is a revised and expanded version of an article entitled 'Replica control following 1SR in DRTDBS through best case of transaction execution' presented at the 'ICDIS 2017', IGNTU, Amarkantak, India, November 3-4, 2017.

1 Introduction

Nowadays, the database system is being incorporated into many organizations and is shared among different users (Elmasri, R., 2008). This system is of two types: (i) distributed and (ii) centralized. In a centralized database system, all data items are confining to a single location. On the other hand, the distributed database distributes the data items in different sites. These database sites are connected using the internet/intranet. The primary correctness factor of the centralized and distributed database system is throughput. Thus, researchers have researched on improving the performance of the system (Garcia-Molina et al., 1990). As the database continues to evolve, various systems such as a real-time system, mobile system, and spatial reference system use the database as a back end. Real-time systems often require predictable processing and timely access to data items (Ulusoy, Ö., 1995b). DRTDBS is explicitly designing for the efficient processing of such data items (Shanker, U. et al., 2008).

Timeliness and temporal consistency are the two primary requirements of DRTDBS. Timeliness is express in the form of a deadline, and the scheduler assigns the priority based on the timeline. Missing the deadline of admitted request/RTTs causes massive loss or a few losses in the system. Based on the consequence of failure, RTT is od three types: (i) hard RTT, (ii) soft RTT, and (iii) firm RTT (Ginis et al., 1998; Kao et al., 1994; Ulusoy, Ö., 1995a). Hard RTT possesses a strong requirement for meeting the deadline. Missing the deadline for hard RTT causes serious catastrophic in the system. Soft RTT does not own the substantial condition of meeting the

ISSN: 2233-7857 IJFGCN

deadline. Thus, soft RTT does not abort on missing the deadline. Firm RTT possesses a strong requirement for meeting

the deadline. However, on missing its deadline, firm RTT does not leave any valuables in the system (Wang, F. et al., 2011). Hence, firm RTT abort on missing the period, such that the remaining RTT gets the system resources for processing.

An admitted request/RTT includes a set of operations and operands. An operation is the processing element that works on data value, whereas operands are the real place holder of such data values. The RTT is processed locally or globally that depends on the availability of a data item in the node (i.e., local or global). A Global RTT establishes one coordinator and more than one cohorts. A Coordinator creates the cohorts and assigns the subset of operations to each cohort. All cohorts process the activities and revert the result to the coordinator (Shanker, U. et al., 2008). During RTT processing in the DRTDBS, there exist different non-deterministic factors that make it challenging to meet the RTT deadline timely. Distributed processing, the existence of single version data items, access latency, and limitation of system memory are some of them. The primary non-deterministic factor is distributed processing. The Distributed processing of RTT causes propagation delay and site delay that makes it rigorous to meet the RTT deadline (Shrivastava, P. et al., 2018b). The second non-deterministic factor is the existence of a single version of data items. A Single version data item allows only single RTT to process in the system and remaining RTT to wait for (Shrivastava, P. et al., 2018b) in the waiting queue. Hence, the existence of a single version data item also makes it rigorous to meet the RTT deadline. The Third non-deterministic factor is access latency caused by the memory that makes the RTT severe to meet the deadline. The last non-deterministic factor is the limitation of system memory that cause RTT to wait for processing in the system. Hence, in DRTDBS, these non-deterministic factors need to solve such that timeliness and consistency demand of DRTDBS get easily satisfied.

The data replication technique is one of the options of DRTDBS, which enhances the availability, scalability, fault-tolerance, and reliability of the system (Shrivastava, P. et al., 2018b). This technique extricates the issue of distributed processing, the existence of single version data item, access latency, and limitation of system memory via creating the data replicas in a higher number of sites. The presence of data replicas prevents the distributed processing of global RTT such that meeting the timeliness demand of admitted RTT becomes easy. Since data replicas exist in a greater number of locations, load sharing becomes easy. This load sharing technique allows waiting RTT to process on another available data replica. The main memory-based DRTDBS redresses the problem of access latency and limitation of system memory. In the present paper, we specifically focus on the data replication technique rather than the main memory-based DRTDBS. Due to the advantages offered by the data replication, we have incorporated data techniques in the DRTDBS to extricate the issues of processing global RTT. Although, data replication technique improves the timeliness but compromises in the mutual consistency of the replicated data item. Hence, researchers are researching in the replicated DRTDBS (RDRTDBS) to solve the issue of mutual consistency (i.e., 4W-H). This 4W-H stands for when to replicate, why to replicate, where to replicate, what to replicate, and how to replicate. Among all researches, the majority of researchers have proposed the solution for how to replicate. Replication protocol solves the issue of how to replicate. The replication protocol simultaneously improves the timeliness and mutual consistency of the system.

Existing replication protocol of distributed database systems is of two types: (i) eager and (ii) lazy (Breitbart, Y. et al., 1997). Eager propagation follows strict consistency criteria, whereas lazy distribution follows weaker consistency criteria — these both propagation mechanisms deprived of real-time constraints. Thus, RDRTDBS (Son, S. 1987.) does not use the replication protocol of the distributed database. In RDRTDBS, Researchers have proposed new replication protocols (Andler S.F. et al., 1996; El-Bakry et al., 2012; Gustavsson S. et al., 2005; Gustavsson S. et al., 2004; Kim, Y.K., 1996; Mathiason, G. et al. 2007; Peddi, P. et al. 2002; Said A.H. et al., 2008;

ISSN: 2233-7857 IJFGCN

Salem R. et al. 2016; Shrivastava, P. & Shanker, U.; Shrivastava P. & Shanker U., 2018cc; Shrivastava P. & Shanker U. 2019a; Shrivastava P. & Shanker U. 2019b; Shrivastava P. & Shanker U. 2020; Son, S., 1987; Son, S.H. & Kouloumbis S., 1993; Son, S.H. & Zhang, F. 1995; Son S.H. et al., 1996; Srivastava A., &Shankar U.; Syberfeldt, S., 2007; Xiong M. et al., 2002) which simultaneously satisfies mutual consistency and timeliness. These replication protocols (Andler S.F. et al., 1996; El-Bakry et al.,

2012; Gustavsson S. et al., 2005; Gustavsson S. et al., 2004; Kim, Y.K., 1996; Mathiason, G. et al. 2007; Peddi, P. et al. 2002; Said A.H. et al., 2008; Salem R. et al. 2016; Shrivastava, P. & Shanker, U.; Shrivastava P. & Shanker U., 2018c; Shrivastava P. & Shanker U. 2019a; Shrivastava P. & Shanker U. 2019b; Shrivastava P. & Shanker U. 2020; Son, S., 1987; Son, S.H. & Kouloumbis S., 1993; Son, S.H. & Zhang, F. 1995; Son S.H. et al., 1996; Srivastava A., &Shankar U.; Syberfeldt, S., 2007; Xiong M. et al., 2002) neither prevents the occurrence of covert channel nor prevents the unauthorized access.

Thus, embedding security in the RDRTDBS becomes an essential part of the research for the researchers. Although incorporating security becomes most necessary in the current scenario, but simultaneously satisfying real-time requirements with security are the most challenging issue in the RDRTDBS. The factors responsible for such an argument is as follows.

1. Limitation of database kernel code extendibility- In RDRTDBS, database kernel broadly possesses the work of concurrency control, buffer management, scheduling the RTT, and replica management. Database kernel embeds replication logic to manage replica management. This logic is tightly coupled with the concurrency control and commit protocol. Hence, it is not easy to easily modify the replication logic. Altering the replication logic affects the performance of concurrency control and vice versa. Thus, a solution is required that brings independence between replication logic and concurrency control or replication logic and commit protocol. Additionally, extending the database kernel with new code is also most rigorous. Thus, extending the database kernel with security policy is the most challenging issue in the RDRTDBS (Shrivastava P. & Shanker, U. 2018b).
2. Existence of covert channel- This channel causes the indirect flow of information between conflicted low clearance level and high clearance level RTTs. The existing security model (i.e., bell Lapadula) partially secures the system and does not prevent the indirect flow of information. Thus, preventing the occurrence of the covert channel in the RDRTDBS is another challenging issue in the RDRTDBS (Son S.H. & Thuraisingham, B., 1993).
3. Existence of external security threats. In RDRTDBS, plain text propagates as an update message in the network. This communication of plain text allows the attackers to trap the message and read the content easily. Similarly, an unauthorized user can modify the message and send the modified signal to the original recipient. Hence, securing the message is also most essential in RDRTDBS such that unauthorized use does not cause harm in the system (Shrivastava, P. et al., 2014).

In RDRTDBS, these mentioned issues need to be solved such that security and real-time constraint get satisfied. In the present paper, our proposed solutions address the question of enforcing security to prevent covert channels and unauthorized access. However, existing solutions offered in (Shrivastava P. & Shanker U., 2018a) solve the issue w.r.t limitation of database kernel code extendibility.

The rest of the paper is as follows. In section 2, the security model discusses unauthorized access inside the system via the covert channel and outside the system via the network. Section 3 presents the solution for open factors such as limitation for kernel code extendibility, the existence of covert channel, and the existence of external security threats. The experimental setting and results are discussed and presented in section 4. Related work is analyzed and presented in section 5, and Finally, in section 6, conclusion and future scope are presented.

ISSN: 2233-7857 IJFGCN

2 Security Model

Due to the evolution of database technology, various real-time applications such as financial services, online stock trading, air traffic control system and soon uses RDRTDBS as a back end for storing their data value. In these real-time applications, information propagates hierarchically. During information propagation, it is necessary to secure the data from unauthorized access such that illegal insider and outsider cannot access the data, and the system remains in a safe state. Existing research work conducted for embedding security

policy in the DRTDBS shows promising results, whereas research towards RDRTDBS is zero. Hence, in RDRTDBS, our objective is to enforce security policy such that simultaneously security, mutual consistency, and timeliness is satisfied. Specifically, we aim to prevent covert channels inside the system and unauthorized access outside the network.

2.1 Covert Channel Inside the System

Existing security policies for DRTDBS are of two types (i)multilevel security policy and (ii) discretionary policy. A discretionary security policy verifies the type of access and the identity of the user to prevent unauthorized access. However, such a strategy cannot prevent the unauthorized disclosure of information. Thus, the system rarely uses this policy. On the other hand, a multilevel security policy is the most frequently used security model in the DRTDBS (Ebrahim Abduljalil, D., 2017). Bell Lapadula is a multilevel security model that consists of objects and subjects (Bell, D.E., LaPadula, L.J., 1973). This model is most frequently used to prevent unauthorized access. In Bell Lapadula security model, objects represent the record or file or data, whereas subjects represent the process. This model consists of two restriction policies named * property and simple security property, which prevents the direct flows of information from high clearance level RTT to low clearance level RTT. The bell Lapadula model offers more security but suffers from the occurrence of the covert channel in the system. A covert channel occurs when different processes have different clearance levels and the conflict on the same data. This conflict causes only one RTT to work and others to wait. Due to the non-availability of data waiting, RTT gets delayed. This presence of delay causes processing RTT to encode information and pass on to another clearance level RTT. Thus, preventing the occurrence of a covert channel is necessary to secure the RDRTDBS.

To prevent the occurrence of the covert channel, high clearance level RTT should never delay the processing of low clearance level RTT. Similarly, value accessed by the low clearance level should not be changed by the high clearance level RTT. Hence, to secure the system from covert channel following properties need to be satisfied (David, R., Son, S.H., Mukkamala, R., 1995).

1. Delay Security- In delay security, a high clearance level RTT should never delay the low clearance level RTT to process in the system.
2. Value Security- In value security, the value accessed by the low clearance level RTT should not be changed by the high clearance level RTT.
3. Recovery Security- When delay security and value security are taken together in the system, it is called deadlock.

As already mentioned in our previous section that security, timeliness, and mutual consistency becomes the primary requirement of RDRTDBS. However, simultaneously satisfying all these properties is not easy in the RDRTDBS because, in DRTDBS, there is always a trade-off between security and timeliness. Hence, meeting mutual consistency, timeliness, and safety is also most challenging in the RDRTDBS.

In the rest of this section, we have presented different scenarios that report when timeliness and security will be satisfied or compromised in the DRTDBS. This scenario is designed based on the priority value and clearance level of RTT. This scenario helps in developing the solution for

ISSN: 2233-7857 IJFGCN

RDRTDBS that simultaneously satisfy mutual consistency, timeliness, and security. For simply understanding these scenarios, we have considered only two RTTs. However, in real-time, there are several RTTs.

Let us consider first RTT R1 possesses priority P1 and clearance level L1. Similarly, second RTT R2 own priority P2 and clearance level L2. T1 and T2 represent the time instance and T1<T2.

1. Scenario 1- If P1 > P2 and L1 < L2, then the result for processing R1 and R2 at different time instances are shown in table 1.

Table 1. Scenario 1

tion			iness	ity

In this scenario, the scheduler must prevent the condition 2nd such that both timeliness and security get satisfied.

2. Scenario 2- If P1 < P2 and L1 > L2, then the result for primarily processing R1 and then R2 is shown in table 2.

Table 2. Scenario 2

tion			iness	ity

In this scenario, the scheduler must prevent the condition 1st such that both timeliness and security get satisfied.

3. Scenario 3- If P1 > P2 and L1 > L2, then the result for processing R1 at R2 at different time instances is shown in table 3.

Table 3. Scenario 3

tion			iness	ity

In this scenario, there is always a trade-off between security and timeliness. Thus, based on user demand scheduler may prefer either case 1st or case 2nd.

4. Scenario 4- If P1 < P2 and L1 < L2, then the result for processing R1 at R2 at different time instances are shown in table 4.

Table 4. Scenario 4

tion			iness	ity

In this scenario also there is always a trade-off between security and timeliness. Thus, based on user demand scheduler may prefer either case 1st or case 2nd.

5. If P1 == P2 and L1 > L2, then the result for processing R1 at R2 at different time instances is shown in table 5.

Table 5. Scenario 5

t ion			iness	ity

In this scenario, the scheduler must prevent condition 1st such that both timeliness and security get satisfied.

ISSN: 2233-7857 IJFGCN

6. Scenario 6- If P1 > P2 and L1 == L2, then the result for processing R1 at R2 at different time instances are shown in table 6.

Table 6. Scenario 6

tion			iness	ity

In this scenario, the scheduler must prevent the condition 2nd such that both timeliness and security get satisfied.

7. Scenario 7- If P1 == P2 and L1 < L2, then the result for processing R1 at R2 at different time instances is shown in table 7.

Table 7. Scenario 7

tion			iness	ity

In this scenario, the scheduler must prevent the condition 2nd such that both timeliness and security get satisfied.

8. Scenario 8- If P1 < P2 and L1 == L2, then the result for processing R1 at R2 at different time instances is shown in table 8.

Table 8. Scenario 8

tion			iness	ity

In this scenario, the scheduler must prevent the condition 1st such that both timeliness and security get satisfied.

9. Scenario 9- If P1 == and L1 == L2, then the result for processing R1 at R2 at different time instances are shown in table 9.

Table 9. Scenario 9

tion			iness	ity

In this scenario, the scheduler may select condition 1st or condition 2nd. In both cases, timeliness and security get satisfied.

From the above set of scenarios, it is clear that trade-off between real-time constraint and security occurs only scenario three and scenario 4. Hence, in the Solution for Covert Channel section, we have proposed the solution for scenario three and scenario 4.

ISSN: 2233-7857 IJFGCN

2.2 Unauthorized Access Outside the System

In RDRTDBS, the replication protocol maintains replica consistency. Replication protocol propagates the replica message in between the master site and slave site/ master site such that data replica consistently reaches to the current state. Since these messages spread in the network in the form of plain text where attackers or unintended users are present to trap the message, an attacker or unwanted user can easily read or modify the message content for which an attacker does not privileges to access the data. Thus, to secure the replica message from such users, it is necessary to use the security technique in the system. Stenography, hash function technique, and cryptography technique are an existing security technique of network security that secures the message communicating int the network. In the stenography technique, the image holds the information, and the sender propagates such an image in the system. The receiver receives such an image and dislodges original data from the image. Cryptography technique converts the original data into an

unreadable format such that unintended user is unable to retrieve the original data (Das S. et al., 2011; Forouzan, B.A., 2007; Li X. et al., 2008; Santhi, B. et al., 2012).

The cryptography technique uses a key and encryption/decryption algorithm to encrypt the original message into an unreadable format. An encrypted message propagates in the network such that unauthorized user is unable to retrieve the original message. In contrast, the receiver decrypts the encrypted message and recovers the original information to process in the system. Both encryption and decryption algorithms handle the substitution and transposition technique on the original message to convert into an unreadable format. Substitution replaces each letter of the message with a new letter, whereas transposition interchanges the position of each letter. These techniques take the input of the key and original message which is provided by the user. In the encryption/decryption algorithm, a smaller key is easily crackable, whereas the key with the bigger size is more challenging to crack. Thus, in cryptography, the key is being securely shared between the sender and receiver (Shrivastava P. et al., 2014).

In cryptography, the encryption algorithm is given by,

$$CT = E(KEY, PT) \quad (1)$$

whereas the decryption algorithm is given by,

$$PT = D(KEY, CT) \quad (2)$$

In equations 1 and 2, CT stands for ciphertext, E stands for the encryption algorithm, D stands for decryption algorithm. And PT stands for plain text.

To improve the security of the message propagating in the network of RDRTDBS, we use the cryptography technique which converts our replica message spreading in the system in an unreadable form such that unintended user is unable to retrieve the original message.

3 Solution for Open Factors

As already mentioned, embedding security policy in RDRTDBS is the most challenging issue. There exist different factors that create difficulty in simultaneously satisfying the timeliness, mutual consistency, and security. In this section, we have proposed the solution for such identified factors.

ISSN: 2233-7857 IJFGCN

3.1 Solution for Database Kernel Code Extendibility

As already mentioned, extending the database kernel with new code is very rigorous in the RDRTDBS. Since replica management and concurrency control, replica management, and commit protocol is related to each other, modification conducted in the concurrency control or commit protocol will impact the performance of replica management. To reduce such impact, we use the middleware (Shrivastava, P. & Shanker U., 2018a) that shifted a load of replica management from the database kernel to the external location. This existing middleware consists of three sub-layers named as data analyzer, conflict detection & correction, and propagation. The data analyzer decomposes the admitted RTT into set operands and operations. This set is forward to the conflict sublayer, which checks the conflict between existing RTT and newly admitted RTT. Conflict detection & correction then schedules the RTT and forward for propagation in the network via the propagation sublayer. The propagation sublayer broadcasts the write/update RTT and unicasts the read RTT to the master site and slave site, respectively.

This middleware isolates the location for RTTs processing in different sites. Write/update RTTs are processed in the master site, whereas the slave site process read RTTs. This isolation eventually increases the performance in terms of transaction miss ratio (TMR). The authors are requested to

read the paper (Shrivastava, P. & Shanker U., 2018a) to get more detail. Figure 1 shows the block diagram of the existing middleware.

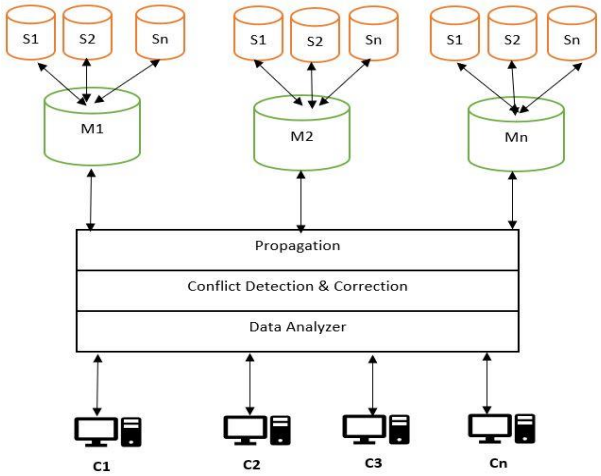

Figure 1. Existing Middleware

In Figure 1, S1, S2, and Sn represented the slave sites. Slave site processes only read RTTs. M1, M2, and Mn represent master sites. Master processes only write and update RTT to update the value of real-time and non-real time data item. C1, C2, and Cn represent clients. Clients are the real service users who submit the request to the master/slave site and receive the response from such a site. Middleware is the interface between master and client.

3.2 Solution for Covert Channel

As already mentioned in our previous section, the covert channel possesses the indirect flow of information between low clearance level RTT and high clearance level RTT. To prevent the occurrence of such channels in the RDRTDBS, we have updated the middleware (Shrivastava, P. & Shanker U., 2018a) with security policy. This sublayer assigns the deadline, priority, clearance

ISSN: 2233-7857 IJFGCN

level, and process the appropriate measure to satisfy the timeliness, mutual consistency, and security. Figure 2 shows an updated version of the middleware with a security mechanism.

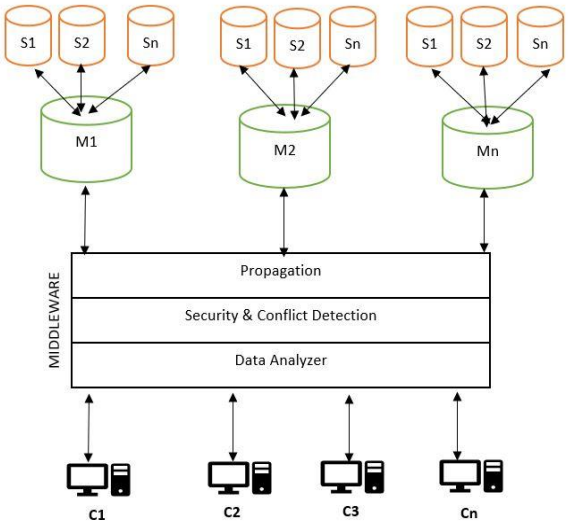

Figure 2. Updated Middleware With Security Constraint

The existing paper (Shrivastava, P. & Shanker U., 2018a) contains the description of data analyzer and propagation. However, the description of the security & conflict detection sublayer and the code working in this sublayer is as follows.

3.2.1 Security & Conflict Detection Sublayer

Primarily, based on the RTT requirement, this sublayer assigns the deadline, priority, and clearance level. After processing initial work, the conflict is checked between admitted RTT and existing RTT, and based on the conflict, appropriate conflict resolution gets initiated such that timeliness, mutual consistency, and security get satisfied. The necessary measure means scheduling the RTT in such a way that all system requirement remains satisfied. This scheduled RTT is then carry over to the propagation sublayer. Propagation sublayer broadcast or unicast the RTT. This broadcasting or unicasting will depend on the RTT type.

According to the earliest deadline policy in a non-overload environment of RDRTDBS, priority assignment to an admitted RTT is inversely proportional to its deadline. That means, RTT with the most rapid period, will be assigned the highest priority. Thus, in our system, the most top priority is attached to the earliest deadline RTT and the lowest priority to long-duration RTT. This assignment policy satisfies the timeliness demand of RDRTDBS. The deadline assignment and priority assignment (Ulusoy, Ö., 1994) for RTT is as follows.

$$DL=AT+ST*RT \quad (3)$$

In equation 3, DL stands for the deadline, AT stands for arrival time, ST stands for slack factor value, and RT stands for resource time.

Similarly, the priority assignment for admitted RTT is as follows.

$$P=C/ (PET-S) \quad (4)$$

In equation 4, C represents criticalness, S represents slack value, and PET represents process execution time. PET consists of t1, t2, and t3. In PET, t1 stands for priority assignment delay, t2 stands for processing delay, and t3 stands for I/O delay. The equation for PET is as follows.

ISSN: 2233-7857 IJFGCN

$$PET=t1+t2+t3 \quad (5)$$

The assignment of security level to an admitted RTT will depend on the RTT requirement. Zero value corresponds to the low clearance level, and the highest non-zero value corresponds to the highest clearance level.

3.2.2 Measure for Preventing Covert Channel

As already mentioned, in our subsection (i.e., Covert Channel Inside the System) of Security Model that satisfying timeliness, security with mutual consistency in scenarios 1,2,5,6,7,8, and 9 is not a big issue. However, adequate protection with real-time requirements in scenarios 4 and 5 is the most challenging issue. This challenge possesses because of a trade-off between security and timeliness in the DRTDBS. Thus, in the existing system (Ahmed Q.N. & Vrbsky, S.V., 1998; Andler, S.F. et al., 1996; David, R. 1995; George B. & Haritsa, J., 1997; George B. & Haritsa, J.R. 2000; Park, C. & Park, S., 1996; Son S.H., 1997; Yingyuan X., 2006), the scheduler may opt for either security or timeliness to maintain either the security or timeliness.

To solve such a trade-off in RDRTDBS, our updated system model is most helpful. Our updated system model consists of master sites, slave sites, middleware, and clients. A Middleware prepares a standard

schedule for all master sites and a usual plan for all slave sites. A schedule prepared for the master site consists of the only write and update RTTs, whereas a schedule ready for the slave site

consists of only read RTTs. Since write and update the RTT process on non-real time and real-time data items, respectively, the occurrence of conflict is NIL. Hence, in the master site, it is not necessary to check the security and conflict in between update and write RTTs, and these RTTs are scheduled based on their priority value.

In our updated system model, read RTT processes in the slave site. In the slave site, read RTT possesses a shared lock to process on the data item. A shared lock simultaneously allows the number of RTTs to handle the same data item. Thus, the number of read RTTs process on the required data item without checking the conflict and security level. Hence, security and timeliness demands get satisfied in the RDRTDBS. However, during RTT processing in the slave site, mutual consistency gets compromised in favor of timeliness and security. This trade-off occurs because RTT updating real-time and non-real time data item is assigned a low priority. Thus, in our updated system model, there is a trade-off between mutual consistency and timeliness/security.

Let us consider a scenario of RDRTDBS where the system possesses N number of RTTs. These RTTs set consist of different types of RTTs, such as update RTT, write RTT, and read RTT. Later, a new RTT is admitted in the system, then how our update system model processes the newly admitted RTT is as follows.

For simple understanding, we consider two instant T1 and T2. At time instant T1, we consider some RTTs are processing in our system. At time instant T2, a new RTT is admitted to our system. For this newly admitted RTT, how our middleware operates is as follows.

1) As already mentioned, at time instant t1, we consider only 4 RTTs exist in our system. These RTTs are TU1, TU2, TW1, and TW2 where TU1 and TU2 represent the update RTTs and TW1, TW2 represents the write RTT. The set of operations of these RTTs is as follows.

ISSN: 2233-7857 IJFGCN

TU1 - R(RT1), W(RT1)
TU2 - R(RT2), W(RT2)
TW1 - R(NRT1), W(NRT1)
TW2 - R(NRT2), W(NRT2)

These RTTs possess the priority PU1, PU2, PW1, PW2, where PU1 > PW1 > PW1 > PU2. The clearance level for these RTTs is LU1, LU2, LW1, LW2 and LU1 > LU2 > LW1 > LW2. Since these RTTs process on different data items, the issue for not satisfying security and timeliness does not occur in the system. Hence, EDF is used to schedule the TU1, TW1, TW2, TU2.

2) Now, consider that at time instant t2 RTT is admitted. Thus, how our middleware process such admitted RTT will depend on the existing RTT type.

T2.1. If update RTT is admitted, then middleware check for priority value and schedule the RTT based on its priority value (i.e., EDF).
For instance, if TU3 is admitted, then the propagation layer broadcast such accepted RTT to all the master sites such that all master sites consistently reach to the common state.
T2.2. If write RTT is admitted in the system, then middleware check for priority value and schedule the RTT based on its priority value (i.e., EDF).
For instance, if TW3 is admitted, then the propagation layer broadcast such admitted RTT to all the master sites such that all master sites consistently reach to the common state.
T2.3. During the admittance of read RTT, there is a trade-off between timeliness/security and mutual consistency. Thus, if read RTT is admitted, then middleware checks for data conflict in between processing RTTs and admitted RTT. If admitted RTT is not conflicted with any processing RTT, then

such RTT is scheduled via EDF technique such that mutual consistency, timeliness, and security get satisfied. However, if admitted is conflicted with processing RTT, then admitted RTT might be scheduled to meet either strict consistency or timeliness and security.
For instance, if TR1 is admitted, then based on user demand, two cases will occur.

Case 1: If user demand for strict consistency criteria, then RTT updating real-time and non-real time data item in the slave site assigned high priority such that read RTT process on the consistent value and generates consistent value.

Case 2: if user demands for weaker consistency criteria, then RTT updating real-time and non-real time data item in the slave site assigned lowest priority such that read RTT process on the inconsistent value and generates inconsistent value.

Thus, during admittance of write RTT and update RTT, it is not necessary to process conflict detection and clearance level check to satisfy mutual consistency, timeliness, and security. An EDF is used to schedule RTTs. However, during the admittance of read RTT, the user requirement is checked. If the user demands strict consistency, then RTT for updating real-time and non-real time data item in the slave site assigned the highest priority such that timeliness and security get compromised in favor of mutual consistency. In contrast, if the user demands timeliness and security, then RTT updating real-time and non-real time data items will be assigned lower priority in comparison to read RTT such that mutual consistency gets compromised in favor of timeliness and security.

3.3 Solution for Unauthorized Access

Enforcement of security in the system is conducted in two parts: (i) Inside and (ii) Outside. Inside security is necessary to prevent the covert channel such that low clearance level RTT neither gets delayed nor its data value gets changed by the high clearance level RTT. However, enforcing security from the outside is more challenging than inside. Since in the outside, the location of the

ISSN: 2233-7857 IJFGCN

unauthorized user is unknown, and such users can attack the system from any side. Thus, enforcement of security from outside the system is most challenging than inside in the RDRTDBS. In the present paper, we use the cryptography technique to secure the message propagating in the network.

In cryptography, the security of the message propagating in the network partially depends on the encryption/decryption algorithm and partly depends on the key. An algorithm is publicly available;
however, the key is being secret and is known to only sender and receiver. However, the key is made confidential and is known to only the sender/receiver. This key exists in the form of alphanumeric code. This form can be easily remembered and can be easily leaked by the receiver or sender. To secure the message from such a leak, we use the key generation algorithm (Shrivastava, P. et al. 2014). The crucial existing generation algorithm uses the 2D image to generate the key. The main advantage of this proposed work is that the sender or receiver does not need to remember the core in mind. The algorithm makes this key. Hence, the point of cheating that the sender has leaked the key or receiver has leaked the key does not arise. However, in the present paper, we use the audio clip to input in the key generation algorithm and generate the key. Audio clip in more secure then images because the image can be easily identified and easy to remember in mind. However, audio cannot be easily visualized; thus, in the present paper, we use an audio file to generate the key.

3.3.1 Key Generation Algorithm
The key generation algorithm takes the input of an audio file in the Wav format and converts the audio file into a byte array. From this byte array, byte values are retrieved from the prime number of locations. These location values are XORed to generate the key value. This key value is shared among middleware, master site, slave site, using asymmetric key cryptography. In RDRTDBS, different participants use the same shared key to encrypt and decrypt the admitted RTT and response. The steps to generate the key is given by,

STEP 1- User record the audio in the WAV format
STEP 2- The recorded sound is converted into a byte array.
STEP 3- Byte value from the prime number locations is retrieved.

STEP 4- These values are XORed into a single value.
STEP 5- The encryption/decryption algorithm uses a single
Value to encrypt and decrypt the message.

Algorithm 1. Key Generation Using Audio File.

Generated key value is given by,

$$KV=BV(1) + BV(13) + BV(29) + BV(43) + BV(79) \quad (6)$$

Where KV stands for key-value, BV (1), BV (13), BV (29), BV (43), and BV (79) holds for byte value at locations 1, 13, 29, 43, and 79.

3.3.2 Encryption Algorithm
Encryption algorithm encrypts the inputted RTT and forwards to the receiver such that unintended user is unable to retrieve the original data. In figure 3, the user inputs an audio and an RTT in the system. Inputted audio is forward to the key generation algorithm. This algorithm generates the key from the sound and forwards the generated key value to the encryption algorithm. A generated key value is hidden from the user. Thus, point for cheating that key value is a leak by the sender or receiver does not arise in the system. A Generated key value and a user admitted RTT is forward

ISSN: 2233-7857 IJFGCN

to the encryption algorithm to encrypt the RTT and forward the encrypted message to the middleware. Since admitted RTT is encrypted from the user site, the unauthorized user is unable to retrieve the original message in the network. In addition to this, the user transfers the generated key to the middleware, master site, and slave site using asymmetric cryptography such that all sites utilize the key value to encrypt and decrypt the message.

The Middleware receives the encrypted message and key-value from the user. Decrypt the received RTT using the key-value provided by the user. Process the RTT in each sublayer, encrypt the message and forward it to the master/slave site. Master site decrypts the receive

encrypted RTT, process the decrypted text in the system, and revert the response in encrypted form to the user. Similarly, if the admitted RTT is read RTT, then the master site forwards the encrypted RTT to the slave site for processing. The slave site decrypts the received RTT, process the decrypted text in the system, and revert the result in the encrypted form to the user. Overall, during communication between user and middleware, middleware and master site, master, and slave site is conducted in an encrypted format.

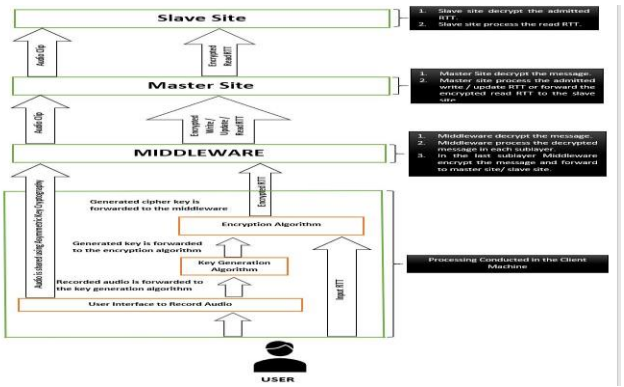

Figure 3. Encryption Algorithm

3.3.3 Decryption Algorithm

The decryption algorithm decrypts the encrypted RTT and forwards it to the receiver such that the user can retrieve the original data. In figure 4, the encrypted response is received from the master and slave site via the middleware. The user passes the recorded audio to the key generation algorithm to generate the key. Generated key and encrypted message is a pass to the decryption algorithm to decrypt the message to retrieve the original response.

To secure the RDRTDBS from the outside, our proposed encryption algorithm, decryption algorithm, and key generation algorithm interactively process together to save the system from massive loss.

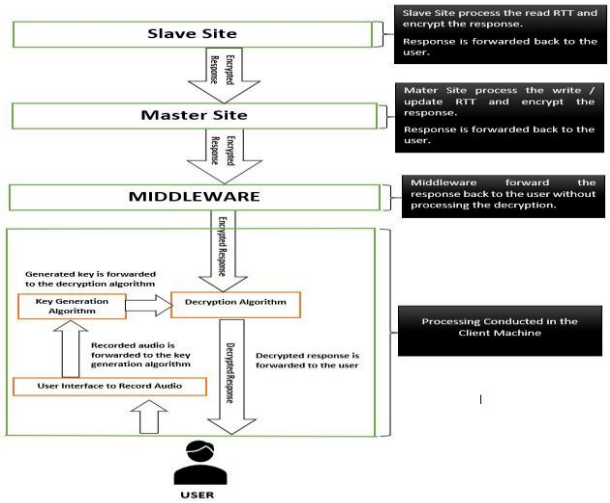

Figure 4. Decryption Algorithm

ISSN: 2233-7857 IJFGCN

4 Experimental Setting & Result

In this section, we have primarily presented the experimental setting and secondary a preliminary result. To measure the performance of our proposed solutions, we have implemented our proposed solutions in the java programming language and compared the effect with an existing solution (Srivastava, A. & Shankar, U.; Xiong M. et al. 2002) of RDRTDBS. Both replication protocols use lock techniques to allows the processing and waiting of RTTs. Lock manager present in the lock technique permit only single RTT to process and remaining RTTs to wait in the waiting queue. Thus, the probability of an occurrence of inconsistency between data items is NIL. Our updated system model also uses lock technique, which allows the processing of only single RTT and remaining RTTs to wait until the lock manager grants the lock for processing on the data item.

During experiment conduction, we have computed the cost of various parameters and based on the result. We can argue that our proposed security solution is beneficial for RDRTDBS.

4.1 Experimental Setting

Simulating settings consist of designing an experimental model and conducting experimentation on such a model. In the present paper, we have developed the empirical model in the java programming language. The java programming language is platform-independent and consists of a rich library that provides us massive support in designing the experimental model. The Netbeans IDE is used to write the java code because it is freely available and offers enormous support for database connectivity, cloud connectivity, and soon.

In our updated system model, replication logic is embedded in the middleware. Hence, the central role is played by the middleware in the experimental model. This middleware is implemented in the java language and placed in a separate system. The middleware takes the encrypted RTT from the user, decrypts the RTT, processes in each sublayer, encrypt the processed RTT, and forward it to the master site (i.e., database). Similarly, when the master site process the admitted RTT, encrypt the response, transmits the encrypted response to the middleware, middleware forward the encrypted response to the user. The user decrypts the response and retrieves the original message. In the experimental model, we have created two master sites and two slave sites for each master site.

4.1.1 Performance Metric for Covert Channel

The computation cost for the covert channel is represented by the performance metrics such as the slave security metric, master security metric, and average security metric.

1) Slave Security Metric- This metric is used to calculate the number of read RTTs completed in their deadline with following security constraints. The slave security metric is given by,

$$SSM=TNRS / TNRR \quad (7)$$

Where SSM stands slave security metric, TNRS stands for the number of read RTTs completed in their deadline with following security constraint, and TNRR stands for the total number of read RTTs admitted in the system.

2) Master Security Metric-This metric is used to calculate the number of write/update RTTs completed in their deadline with following security constraint.

$$MSM=TNS / TNUWR \quad (8)$$

Where MSM stands master security metric, TNS stands for the number of write/update RTTs completed in their deadline with following security constraint, and TNUWR stands for the total number of write/update RTTs admitted in the system.

ISSN: 2233-7857 IJFGCN

3) Average Security Metric- This metric is calculated by the middleware from the data collected by all master sites.

ASM=(MS1+MS2+...+MSn)/TNMS (9)

MS1, MS2, ..., MSn stands for master site1, master site2, ..., master site n. TNSM stands for the total number of master sites.

4.1.2 Performance Metric for Covert Channel

The Performance metric for cryptography is represented in terms of encryption time, decryption time, the throughput of encryption, throughput of decryption, diffusion analysis, CPU process time, and CPU clock cycles, power consumption and memory utilization (Mr.B.Bharathi et al., 2017)

1) Encryption time- This metric is used to calculate the time elapsed in converting the plain text into ciphertext.

ET=SR/TE2-TE1 (10)

Where ET stands for the encryption time, SR stands for the size of admitted RTT, TE1, and TE2 represent the time instance where and TE2>TE1. Encryption of admitted RTT is initialized at time instant TE1 and at TE2 encryption of RTT get completed. Thus, TE2-TE1 represents the CPU cycles invested in converting admitted RTT into ciphertext.

2) Decryption time- This metric is used to calculate the time elapsed in converting the ciphertext into plain text.

DT=DR/TD2-TD1 (11)

Where DT stands for the decryption time, DR stands for the size of encrypted RTT, TD1, and TD2 represent the time instance and TD2>TD1. The decryption of ciphertext is initialized at time instant TD1, and TD2 decryption of ciphertext gets completed. Thus, TD2-TD1 represents the CPU cycles invested in converting cipher text into plain text.

3) Throughput of encryption- The performance of the encryption algorithm depends on the encryption time. It represents the rate of encryption (Naveen Kolhe, N.R., 2013) and is indirectly related to the power consumption of the system. An increase in throughput decreases the power consumption of the system. Similarly, a decrease in throughput increases the power consumption of the system.

THE=SET/ET (12)

Where THE is the throughput of encryption, SET stands for the size of encrypted RTT, and ET stands for the encryption time.

4) Throughput of decryption- The throughput of the decryption algorithm depends on the decryption time. It represents the rate of decryption and is indirectly related to the power consumption of the system. An increase in throughput decreases the power consumption of the system. Similarly, a decrease in throughput increases the power consumption of the system.

THD=SDT/DT (13)

ISSN: 2233-7857 IJFGCN

Where THD is the throughput of decryption, SDT stands for the size of decrypted RTT, and DT stands for the decryption time.

5) CPU process time- This metric is used to calculate the dedicated CPU time invested in the encryption or decryption of admitted RTT.

CPT=BCE*NCI (14)

Where CPT stands for CPU processing time, BCE stands for essential cost of encryption time, and NCI stands for the total number of clock cycles.

4.1.2 Performance Metric for Admitted RTT

However, the Performance metric for admitted RTT is different and is represented by the restart ratio, average miss ratio, and transaction miss ratio.

1) Restart ratio - This metric calculates the number of RTTs restarted from the total number of RTTs.

RR=NRR/TNR

Where RR stands for restart ratio, NRR stands for the number of RTTs restarted due to data conflict, and TNR stands for the total number of RTTs admitted in the system for service.

2) Transaction miss ratio.

TMR=TNSR/TNR

Where TNSR stands for the number of RTTs successfully completed within their deadline, and TNR stands for the total number of admitted RTTs.

In addition to this performance metric, the general setting and user workload setting is given by,

Table 10. System and User Setting

meter	ue
z	)
Tm	sec
b	
ASL	
	)]
z	, 1KB]
y	c
y	c
k	
k	

In Table 10, the number of data pages and processing time on each data page is represented by DB_Sz and CPU_Tm, respectively. WR_Pb represents the probability of update or write operation. Max_ASL represents the security access level for each RTT. Ar_Rt represents the number of RTT admitted in the system for service and ranges from 2 to 100. Txn_Rt represents the average RTT size, and its value is 10. Rs_Dy and Log_Dy, respectively, represent overhead form restart and log access. Slack factor value is used for each RTT and its minimum amount, and the maximum value is represented by Min_Sk and Max_Sk respectively.

ISSN: 2233-7857 IJFGCN

4.2 Experimental Result

To compute the cost for maintaining security in the RDRTDBS, we have conducted the simulation more than ten times. In most of the experiments, we have to change the user setting and collect the result. From the result, we got the confidence that our proposed solution prevents unauthorized access from inside and outside the system.

The collected result during experimentation for the covert channel admitted RTT and unauthorized access is shown in subsection 1, 2, and 3, respectively.

4.2.1 Computation Cost For Maintaining Inside Security

During experiment conduction, we have varied the arrival rate for update/write/read RTT and admit them in the middleware. The Middleware process the admitted RTT and broadcast or unicast the RTT to the master site or slave site, respectively.

In the slave site, the arrival rate for read RTT gets varied and ranges from 2 to 100. The slave site uses a shared lock for the processing of more than one RTTs on the same item. The factor such as delay security, value security, and recovery security responsible for the occurrence of the covert channel does not occur in the slave site. Although the slave site prevents the presence of the covert channel, there is a trade-off between timeliness, security, or mutual consistency. If the user demands timeliness and security, then mutual consistency gets compromised. Figure 5 shows the experimental result for following weaker consistency criteria. On the other demand, If the user demands strict consistency, then timeliness and security get compromised in favor of mutual consistency. Figure 6 shows the experimental result for the strict consistency criteria.

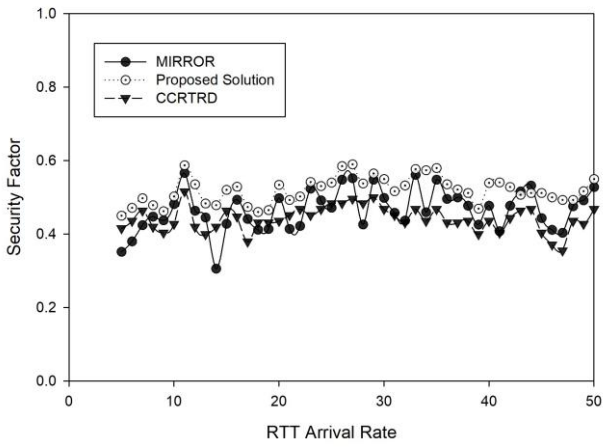

Figure 5. Slave Site Security Under Weaker Consistency Criteria

ISSN: 2233-7857 IJFGCN

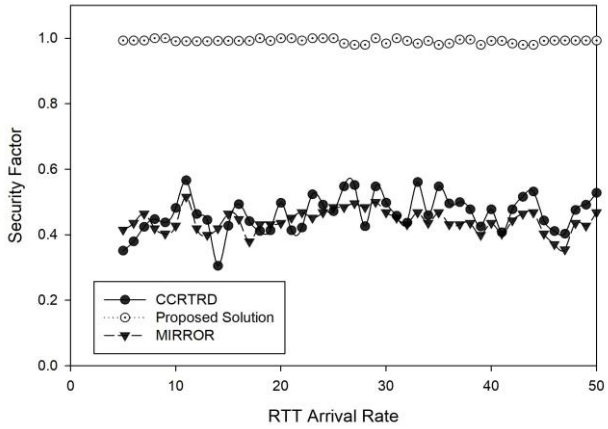

Figure 6. Slave Site Security Under Strict Consistency Criteria

In our updated system model, master site processes write and update RTT. Write and update RTT processes on the different data items. Hence, the occurrence of data conflict between write RTT and update RTT is NIL. Due to the non-occurrence of data conflict, RTT neither suffers from the issue of delay security, value security, and recovery security. Hence, in master site covert channel occurrence, neither occurs. Figure 6 shows the experimental result for the secret channel in the master site.

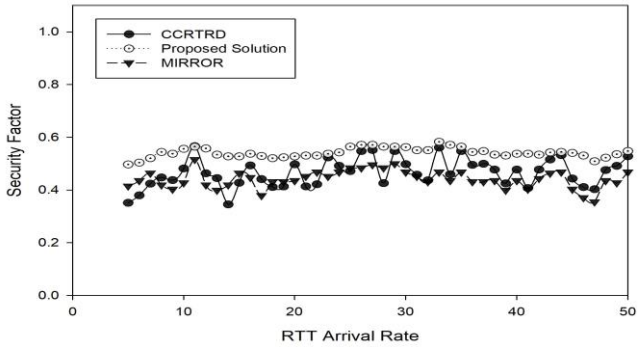

Figure 7. Master Site Security

Since in our updated system model, middleware is the central processing location for all types of RTTs. It periodically collects the performance data from all master sites and calculates the average

security metric. The security metric of individual master sites shows promising performance, and the computed cost from the security metrics of all master sites also shows promising results.

ISSN: 2233-7857 IJFGCN

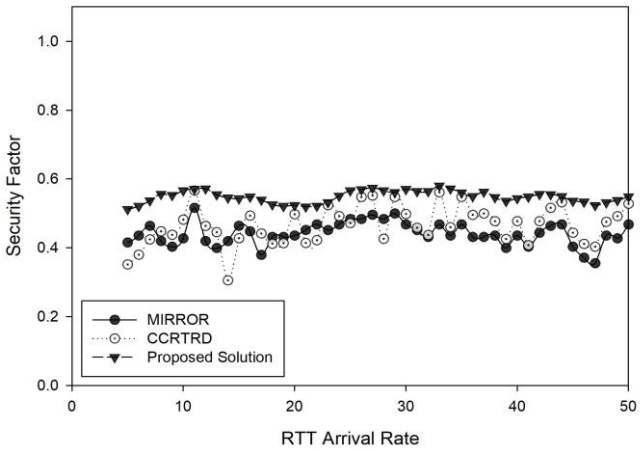

Figure 8 . Slave Site Security Under Strict Consistency Criteria

4.2.2 Performance Metric for Admitted RTT

Despite estimating security performance, computation cost for the admitted RTTs is measured via restart ratio, and TMR. Restart ratio measures the number of RTTs restarted from the total number of admitted RTTs. TMR estimates the number of RTTs misses their deadline due to the occurrence of data conflict or lack of CPU cycles.

To measure the restart ratio, we have admitted the different types of RTTs in our updated system model. From the collected result, we found that in the master site RTTs are restarted because of lack of CPU cycles whereas in the slave site RTTs are restarted because of a trade-off between mutual consistency and timeliness/security. The recorded result for the restart ratio in the master side is shown in figure 9.

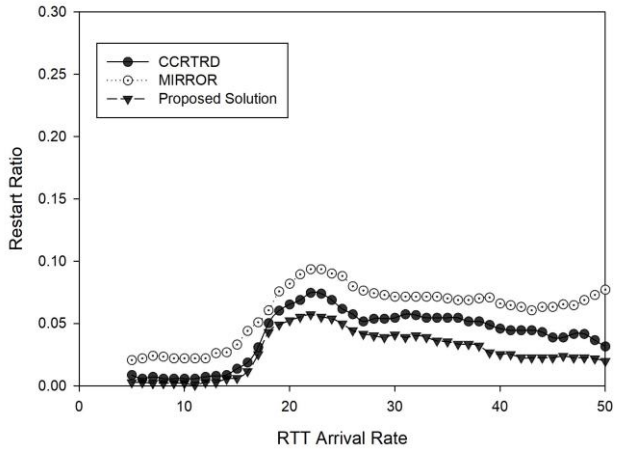

Figure 9. Master Site Restart Ratio

However, in the slave site, if strict consistency is required, then timeliness and security get compromised. Fig.10 shows the experimental result for following strict consistency criteria. On the other hand, if the user demands weaker consistency, then timeliness and security get satisfied. Fig. 11. shows the experimental result for following weaker consistency.

ISSN: 2233-7857 IJFGCN

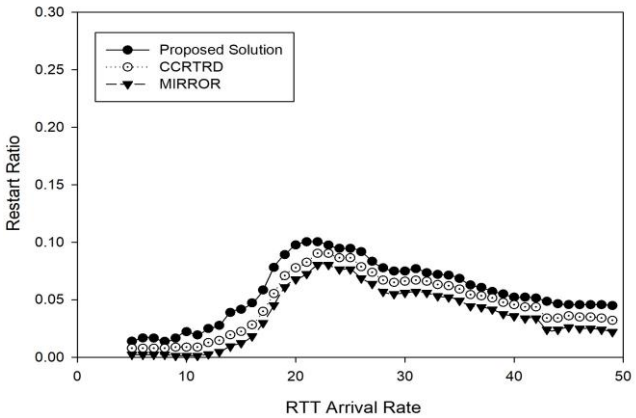

Figure 10. Slave Site Restart Ratio Under Strict Consistency

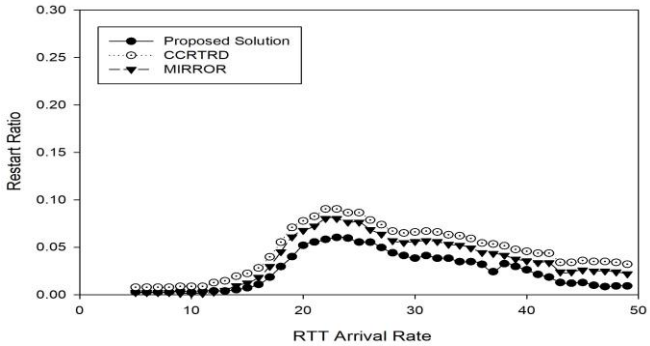

Figure 11. Slave Site Restart Ratio Under Weaker Consistency

4.2.3 Computation Cost For Maintaining Outside Security

In this subsection, we have presented the result collected from the experiment conducted for outside security in our updated system model. The Experimental result is represented in the form of encryption time, encryption throughput, decryption time, decryption throughput, CPU process time, and memory utilization. Figure 12 shows the experimental result for encryption time and decryption time of all RTT, whose size varies from 100 B to 1000 Bytes.

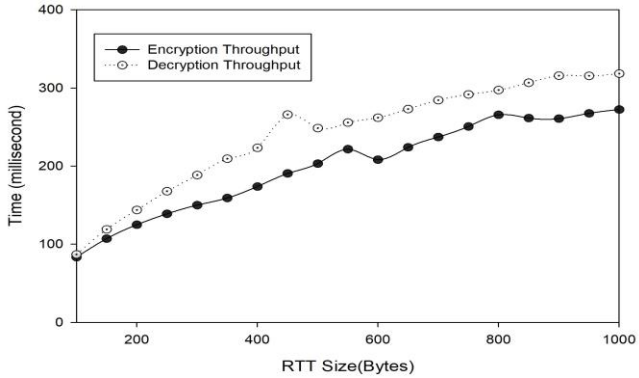

Figure 12. Encryption & Decryption Time

Encryption and decryption throughput is represented in Figure 13.

ISSN: 2233-7857 IJFGCN

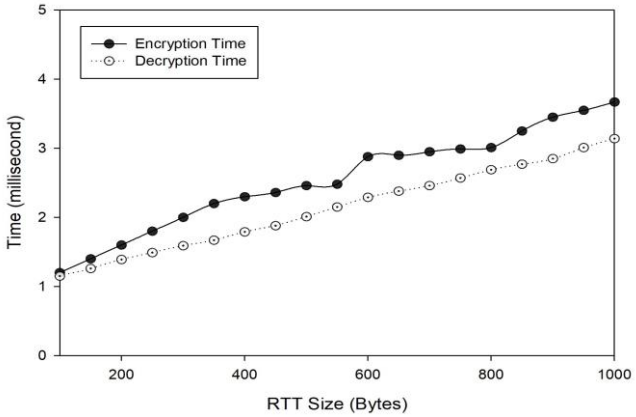

Figure 13. Encryption & Decryption Throughput

The CPU usage is represented in Figure 14.

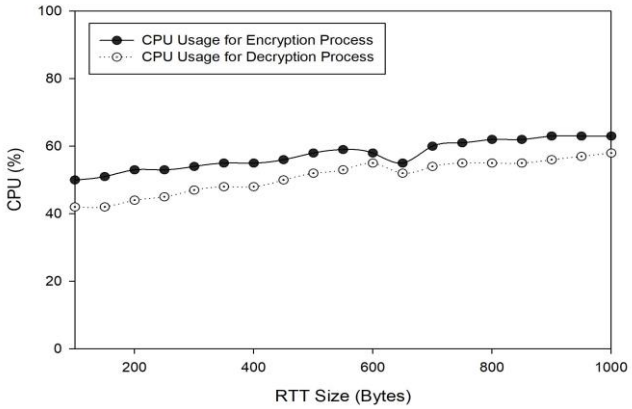

Figure 14. CPU Process Time

5 Related Work

In this section, research in the replication protocol for RDRTDBS and research in security for DRTDBS is presented in subsection 1 and 2, respectively.

5.1 Research in Replication Protocol

In the replicated environment of DRTDBS, the majority of researchers have focused on the development of RPL (Andler S.F. et al., 1996; El-Bakry et al., 2012; Gustavsson S. et al., 2005; Gustavsson S. et al., 2004; Kim, Y.K., 1996; Mathiason, G. et al. 2007; Peddi, P. et al. 2002; Said A.H. et al., 2008; Salem R. et al. 2016; Shrivastava, P. & Shanker, U.; Shrivastava P. & Shanker U., 2018c; Shrivastava P. & Shanker U. 2019a; Shrivastava P. & Shanker U. 2019b; Shrivastava P. & Shanker U. 2020; Son, S., 1987; Son, S.H. & Kouloumbis S., 1993; Son, S.H. & Zhang, F. 1995; Son S.H. et al., 1996; Srivastava A., &Shankar U.; Syberfeldt, S., 2007; Xiong M. et al., 2002) such that RTT processed on the consistent value of the replicated database site. Among such works, research work (Son, S., 1987) exploited the semantic information of read RTT. The experimental result proves that the efficiency of the system gets increases via using such a technique. RPL based on a token scheme and an integrated version of RPL with scheduling is proposed in (Son, S.H. & Kouloumbis, S., 1993; Son, S.H., Zhang, F., 1995), respectively. Both RPLs follow a weaker correctness criterion (i.e., epsilon serializability) for the processing of RTTs in different sites. RPL, based on strict correctness criteria (i.e., 1SR) named as MIRROR (Xiong, M et al. 2002), is an augmented version of O2PL with a novel state-based real-time conflict resolution mechanism.

Similarly, to provide real-time capabilities for the telecom application, a distributed and parallel DBMS named ClustRa is proposed in (Kim, Y.K., 1996). However, these RPLs suffers from the

ISSN: 2233-7857 IJFGCN

issue of an overload condition, unbounded delay, and deadlock. RPL for distributed real-time object

oriented database system is presented in (Peddi, P. & DiPippo, L.C., 2002) and is suited for a static environment.

However, this RPL is not suitable for a dynamic environment where the request is active. In RDRTDBS, conflict detection and correction policy play an essential role in maintaining mutual consistency. Thus, RPL (Gustavsson, S. & Andler, S.F., 2004) conducts continuous conflict detection and correction using the conflict set. This RPL also suffers from the issue of strict consistency criteria. Continuous convergence protocol (Gustavsson, S. & Andler, S., 2005) for DRTDBS concentrates on three main terms (i) local predictability, (ii) local consistency, and (iii) eventual global consistency. However, in such protocol inconsistency for read RTT has to be tolerated. An RPL based on an optimistic approach with deterministic detection and forward resolution of conflicts is proposed in (Syberfeldt, S., 2007). This RPL is known as PRiDe and primarily focuses on maintaining mutual consistency of real-time data items, but RPL (Said, A.H et al. 2008) focuses on maintaining mutual consistency of non-real time data item. The simulator proposed in (El-Bakry et al. 2008) examines the performance of RPLs. Virtual full replication based on adaptive segmentation (Mathiason, G. et al. 2007) resolves the severe drawbacks of full replication, but such an approach also suffers from overloading issues. Recently, the middleware-based replica control technique (MBRCT) following 1SR is proposed in (Shrivastava, P., Shanker, U., 2018a) for increasing the performance and scalability.

5.2 Research in Security for DRTDBS

In the research article (Son, S.H. & Thuraisingham, B., 1993), the author has reported different issues to solve the trade-off between timeliness and security. Partial violation of security is presented in (David, R. et al. 1995), and an extended version of this approach with the adaptive policy is proposed in (Son S.H., 1997). Another solution based on multiple version techniques to solve the trade-off is reported in (Park, C. & Park, S., 1996). An integrated approach to address inter and intra level conflict is proposed in (George, B. & Haritsa, J., 1997). Optimistic concurrency control with security constraint for DRTDBS in introduced in (Ahmed, Q.N. & Vrbsky, S.V., 1998.). SABRE (secure algorithm for buffering in real-time environments) (George, B. & Haritsa, J.R., 2000) provides covert channel free security. A new concurrency control protocol named ROT-FREEZE (Han H. et al., 2000) improves the performance for read RTT. Optimistic based concurrency control protocol (Yingyuan X. et al. 2006) has proposed two factors: (i) secure influence factor and (ii) real-time influence factor. These factors solve the trade-off between security and timeliness. Recently, Ebrahim Abduljalil et al. has published paper (Ebrahim Abduljalil, D., 2017), that details about multi security models in a real-time database.

6 Conclusion

The main requirement for DRTDBS is timeliness and temporal consistency. This requirement becomes more complicated in the presence of various non-deterministic factors such as distributed processing, the existence of single version data objects, access latency, and limitation of system memory. The replication technique extricates the issue for distributed processing and the presence of individual version data objects. A Replication technique easily meets the timeliness demand by creating the same data replica in different locations. The major challenging issue in the replication technique is the satisfaction of replica consistency. Existing researches have been mainly conducted on maintaining replica consistency for RDRTDBS. These researches suffer from the issue of the covert channel, unauthorized access outside the system, and limitation of kernel code extendibility. The solution for limitation of kernel code extendibility is already proposed in (Shrivastava, P., Shanker, U., 2018a), and we have embedded the solution for covert channel and unauthorized access in these solutions such that the occurrence of the secret channel and

ISSN: 2233-7857 IJFGCN

unauthorized access get prevented in the system. The performance of our proposed solution is implemented in java programming and tested with other non-secure replication protocol. From the collected result, we can argue that our proposed solution for inside and outside security in the RDRTDBS prevents the unauthorized access inside as well as outside the system. Our proposed solution is beneficial for real-time application, which demands performance and safety.

References

1. Ahmed, Q.N., Vrbsky, S.V., 1998. Maintaining security in firm real-time database systems, in: Proceedings 14th Annual Computer Security Applications Conference (Cat. No. 98EX217), IEEE. pp. 83–90.

2. Andler, S.F., Hansson, J., Eriksson, J., Mellin, J., Berndtsson, M., Eftring, B., 1996. Deeds towards a distributed and active real-time database system. ACM Sigmod Record 25, 38–51.

3. Bell, D.E., LaPadula, L.J., 1973. Secure computer systems: Mathematical foundations. Technical Report. MITRE CORP BEDFORD MA.

4. Breitbart, Y., Korth, H.F., 1997. Replication and consistency: Being lazy helps sometimes, in: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of database systems, pp. 173–184.

5. Das, S., Das, S., Bandyopadhyay, B., Sanyal, S., 2011. Steganography and steganalysis: different approaches. arXiv preprint arXiv:1111.3758.

6. David, R., Son, S.H., Mukkamala, R., 1995. Supporting security requirements in multilevel real-time databases, in Proceedings 1995 IEEE Symposium on Security and Privacy, IEEE. pp. 199–210.

7. Ebrahim Abduljalil, D., 2017. Multilevel security models in real-time database systems: Comparing and analyzing — International Journal Of Engineering And Computer Science 6.

8. El-Bakry, H.M., Sultan, T., et al., 2012. Design of replicated real-time database simulator, in Proceedings of the 6th WSEAS International Conference on Computer Engineering and Applications, and Proceedings of the 2012 American conference on Applied Mathematics. World Scientific and Engineering Academy and Society (WSEAS).

9. Elmasri, R., 2008. Fundamentals of database systems. Pearson Education, India.

10. Forouzan, B.A., 2007. Cryptography & network security. McGrawHill, Inc.

11. Garcia-Molina, H., Lindsay, B., 1990. Research directions for distributed databases. ACM SIGMOD Record 19, 98–103.

12. George, B., Haritsa, J., 1997. Secure transaction processing in firm real-time database systems. ACM SIGMOD Record 26, 462–473.

13. George, B., Haritsa, J.R., 2000. Secure buffering in firm real-time database systems. The VLDB Journal 8, 178–198.

14. Ginis, R., Wolfe, V.F., 1998. Issues in designing open distributed realtime databases, in Proceedings of the 4th international workshop on Parallel and Distributed Real-Time Systems, IEEE. pp. 106–109.

ISSN: 2233-7857 IJFGCN

15. Gustavsson, S., Andler, S., 2005. Continuous consistency management in distributed real-time databases with multiple writers of replicated data in 19th IEEE International Parallel and Distributed Processing Symposium, IEEE. pp. 8–pp.

16. Gustavsson, S., Andler, S.F., 2004. Real-time conflict management in replicated databases, in: Proceedings of the Fourth Conference for the Promotion of Research in IT at New Universities and University Colleges in Sweden (PROMOTE IT 2004), Karlstad, Sweden, pp. 504–513.

17. Han, H., Park, S., Park, C., 2000. A concurrency control protocol for read-only transactions in real-time secure database systems, in Proceedings Seventh International Conference on Real-Time Computing Systems and Applications, IEEE. pp. 458–462.

18. Kao, B., Garcia-Molina, H., 1994. An overview of real-time database systems, in Real-Time Computing. Springer, pp. 261–282.

19. Kim, Y.K., 1996. Towards real-time performance in a scalable, continuously available telecom DBMS.

20. Li, X., Wu, X., Qi, N., Wang, K., 2008. A novel cryptographic algorithm based on iris feature, in 2008 International Conference on Computational Intelligence and Security, IEEE. pp. 463–466.

21. Mathiason, G., Andler, S.F., Son, S.H., 2007. Virtual full replication by adaptive segmentation, in the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007), IEEE. pp. 327–336.

22. Mr.B.Bharathi, Mr.G.Manivasagam, D.K., 2017. Metrics for performance evaluation of encryption algorithms. International Journal of Advance Research in Science and Engineering 6, 62–72.

23. Naveen Kolhe, N.R., 2013. Throughput comparison results of proposed algorithm with existing algorithm. The International Journal Of Engineering And Science (IJES) 2, 92–98.

24. Park, C., Park, S., 1996. A multiversion locking protocol for real-time databases with multilevel security, in Proceedings of 3rd InternationalWorkshop on Real-Time Computing Systems and Applications, IEEE. pp. 136–143.

25. Peddi, P., DiPippo, L.C., 2002. A replication strategy for distributed real-time object-oriented databases, in Proceedings Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. ISIRC 2002, IEEE. pp. 129–136.

26. Said, A.H., Sadeg, B., Amanton, L., el Ayeb, B., 2008. A protocol to control replication in distributed real-time database systems., in ICEIS (1), pp. 501–504.

27. Salem, R., Abdul-Kader, H., et al., 2016. Scalable data-oriented replication with flexible consistency in real-time data systems. Data Science Journal 15.

28. Santhi, B., Ravichandran, K., Arun, A., Chakkarapani, L., 2012. A novel cryptographic key generation method using image features. Research Journal of Information Technology 4, 88–92.

29. Shanker, U., Misra, M., Sarje, A.K., 2008. Distributed real-time database systems: background and literature review. Distributed and parallel databases 23, 127–149.

ISSN: 2233-7857 IJFGCN

30. Shrivastava, P., Jain, R., Raghuwanshi, K., 2014. A modified approach of key manipulation in cryptography using 2d graphics image, in 2014 International Conference on Electronic Systems, Signal Processing, and Computing Technologies, IEEE. pp. 194–197.

31. Shrivastava, P., Shanker, U.. Predicting processing time of real-time transaction in replicated drtdbs via middleware.

32. Shrivastava, P., Shanker, U., 2018a. Replica control following 1sr in drtdbs through best case of transaction execution, in Advances in Data and Information Sciences. Springer, pp. 139–150.

33. Shrivastava, P., Shanker, U., 2018b. Replica update technique in rdrtdbs: issues & challenges, in Proceedings of the 24th International Conference on Advanced Computing and Communications (ADCOM-2018), Ph. D. Forum, Bangalore, India, pp. 21–23.

34. Shrivastava, P., Shanker, U., 2018c. Replication protocol based on dynamic versioning of data object for replicated drtdbs. International Journal of Computational Intelligence & IoT 1.

35. Shrivastava, P., Shanker, U., 2019a. Real-time transaction management in replicated drtdbs, in Australasian Database Conference, Springer. pp. 91–103.

36. Shrivastava, P., Shanker, U., 2019b. Supporting transaction predictability in replicated drtdbs, in International Conference on Distributed Computing and Internet Technology, Springer. pp. 125–140.

37. Shrivastava, P., Shanker, U., 2020. Secure system model for replicated drtdbs, in Security and Privacy Issues in Sensor Networks and IoT. IGI Global, pp. 264–281.

38. Son, S., 1987. Using replication for high performance database support in distributed real-time systems.

39. Son, S.H., 1997. Supporting timeliness and security in real-time database systems, in Proceedings Ninth Euromicro Workshop on Real-Time Systems, IEEE. pp. 266–273.

40. Son, S.H., Kouloumbis, S., 1993. A token-based synchronization scheme for distributed real-time databases. Information Systems 18, 375–389.

41. Son, S.H., Thuraisingham, B., 1993. Towards a multilevel secure database management system for real-time applications, in: [1993] Proceedings of the IEEE Workshop on Real-Time Applications, IEEE. pp. 131–135.

42. Son, S.H., Zhang, F., 1995. Real-time replication control for distributed database systems: Algorithms and their performance., in DASFAA, pp. 214–221.

43. Son, S.H., Zhang, F., Hwang, B., 1996. Concurrency control for replicated data in distributed real-time systems. Journal of Database Management (JDM) 7, 12–23.

44. Srivastava, A., Shankar, U., Ccrtrd: A protocol for concurrency control in real-time replicated databases system.

45. Syberfeldt, S., 2007. Optimistic replication with forward conflict resolution in distributed real-time databases. Ph.D. thesis. Institutionen för datavetenskap.

46. Ulusoy, Ö., 1994. Processing real-time transactions in a replicated database system. Distributed and Parallel Databases 2, 405–436.

ISSN: 2233-7857 IJFGCN

47. Ulusoy, Ö., 1995a. Research issues in real-time database systems: survey paper. Information Sciences 87, 123–151.

48. Ulusoy, Ö., 1995b. A study of two transaction-processing architectures for distributed real-time database systems. Journal of Systems and Software 31, 97–108.

49. Wang, F., Yao, L.W., Yang, Y.L., 2011. Efficient verification of distributed real-time systems with broadcasting behaviors. Real-Time Systems 47, 285.

50. Xiong, M., Ramamritham, K., Haritsa, J.R., Stankovic, J.A., 2002. Mirror: A state-conscious concurrency control protocol for replicated real-time databases. Information systems 27, 277–297.

51. Yingyuan, X., Yunsheng, L., Xiangyang, C., 2006. An efficient, secure real-time concurrency control protocol. Wuhan University Journal of Natural Sciences 11, 1899–1902.

ISSN: 2233-7857 IJFGCN