An Efficient Backup Technique for Database Systems Based on Threshold Sharing

Sahel Alouneh[1], Sa'ed Abed[2], Bassam Jamil Mohd[2], Mazen Kharbutli[3]
[1]German Jordanian University, [2]Hashemite University, [3]Jordan University of Science and Technology
{sahel.alouneh@gju.edu.jo, sabed@hu.edu.jo, Bassam@hu.edu.jo, kharbutli@just.edu.jo}
Corresponding and principle author: Sahel Alouneh, Ph.D

***Abstract*—Database security, corruption, and loss can be disruptive, time-consuming and expensive to organization operation and business continuity. Therefore, data protection and availability is a high priority and a sensitive concern during the design and implementation of information systems infrastructure. This paper proposes a novel technique for designing and implementing a database recovery and security system based on the threshold secret sharing scheme. Furthermore, a network-based database protection technique is devised and presented. Analysis of the proposed technique shows that it is effective and comprehensive while not imposing significant delay in order to secure, distribute and recover the data.**

***Index Terms*— Database; Threshold sharing; Security; Recovery**

I. Introduction and motivation

Business continuity necessitates the existence of a reliable database system. A database error or failure, even if for a short time period, can cause financial and social confusion, and may lead to the loss of valuable customer confidence [1, 2]. A database system failure may be caused by various reasons such as human errors or hardware failures. To ensure the reliability of the database in a system, the database management and recovery policy must be prepared beforehand prior to the occurrence of such errors and failures. In addition, the security of database content is of great concern, and in some situations is considered the first priority.

This paper proposes a novel and efficient technique for designing and implementing a database protection system that is based on the Threshold Secret Sharing Scheme (TSSS) [3]. One main motivation behind this work is the ability to use the natural characteristics of the threshold secret sharing scheme in order to provide security and recovery for database systems. On the other hand, the original TSSS requires extra overhead for coding the shares and therefore this work suggests modification to the original TSSS to help in reducing this overhead size, especially when using this technique to recover from data loss and protecting the integrity of the data. The proposed technique relies on dividing the database into shares and distributing them to an array of hard drives (Sub-storage databases) in such a way that would make it possible to recover the original database even if not all hard drives are able to provide their shares. That is, the shares would contain enough redundancy making it possible for the whole database to be recovered if enough (but not all) hard drives are able to provide their shares. In addition, the proposed technique is also able to protect the confidentiality and integrity of the database system because the shares are coded. The proposed technique is analyzed and shown to be effective while imposing insignificant delays and overhead.

The rest of the paper is organized as follows. Section 2 discusses the database protection background and related work. After that, Section 3 presents the proposed technique and Section 4 evaluates and compares it with other existing techniques. Finally, Section 5 concludes the paper.

II. Database Protction Background and Related Work

Data recovery is the process of preserving data from damage or destruction. It may be required due to failures or errors that turn up in the form of physical damage to the storage device or logical damage to the file system that prevents it from being mounted by the host operating system.

A primary storage medium used for data storage is the hard disk. Hard disks have higher failure rates relative to other storage mediums necessitating the existence of a backup and recovery system. A simple method that can be used to protect from hard disk failures is to make backup copies of the files on other mediums such as redundant hard disks, magnetic tapes, removable disks, or magnetic optical disks [2].

Currently, a popular method used to improve the reliability and performance of database storage on hard disks is RAID arrays (Redundant Arrays of Inexpensive Disks) [4]. RAID technology was developed to address the limitations and drawbacks of conventional disk storage systems in terms of fault-tolerance and performance. It can offer an improved fault tolerance and higher throughput levels compared to a single hard drive or a group of independent hard drives. While RAID arrays were once considered complex and expensive storage solutions, today they are easy to use, relatively inexpensive, and have become essential for a broad range of applications.

In a RAID solution, multiple drives are organized in a single array, which is viewed by the operating system as a single disk. There are several different RAID "levels" or

doi:10.4304/jcp.8.11.2980-2989

redundancy schemes, each with its inherent cost, performance, and availability (fault-tolerance) characteristics. Each level is designed to meet different user and system requirements. RAID levels 0, 1, and 5 are the most common and cover most requirements. Table I summarizes the main characteristics of these RAID levels. In Table II, a summary of database protection techniques is presented.

Many recovery schemes have been proposed to address various performance issues. The work proposed by Ammann *et. al*. [5] considers the database recovery from a security point of view. It concentrates on database failures caused by attacks that aim to reduce the availability by bringing the system down. To recover from such attacks, the authors developed a family of trusted repair algorithms, each of which has two versions: static and dynamic. The tradeoffs vary between static repair and dynamic protection characteristics in terms of service delay and the amount of recoverable data.

TABLE I.
COMPARISON BETWEEN RAID LEVELS

RAID level	*Level 0*	*Level 1*	*Level 5*
Minimum number of drives	2	2	3
Strengths	Highest performance.	High performance; High data protection.	High performance; High data protection; Lower redundancy cost compared to RAID 1.
Weaknesses	No data protection; If one drive fails, all data is lost.	High redundancy cost overhead.	Moderate redundancy cost overhead; Complexity is more than RAID 0 and 1.

TABLE II.
LIST OF SOME DATABASE PROTECTION TECHNIQUES

Technique	*Strengths*	*Weakness*
Ammann et.al [5]	-Recovery from DoS attacks.	-No data failure recovery. -Variable recovery delay.
Nakamura et. al. [1].	-Partial recovery	-When to apply the full backup. -Variable overhead.
HDFS [4]	-Asynchronous Compression -RAID compatible.	-Large delay overhead
Lui et. al. [15]	-Specialized for image data. -Grid computing	-High overhead.
Lu. et.al [11]	-Hierarchical clustering -Scalable	Reverse clustering
Kadhem et. al. [18] (MCDB)	Protection for Confidentiality, privacy, integrity of data.	No failure recovery
Storer et. al [17]	Protects database for security and failure.	High overhead redundancy and delay

Other database recovery mechanisms aim to protect the database system from malfunctions caused by hardware failures. Nakamura *et. al*. [1] considers the problem of when to make a full backup. They propose an optimal backup policy for a database system with incremental and full backups. The incremental method takes only copies of newly updated files that are usually adopted in most database systems. An important concern when applying this technique is the necessity of knowing when to apply the full backup. Also, the overhead of an incremental backup increases in proportion to the total amount of updated files and the backup process depends on the total amount of newly updated files.

The work presented by Kawai *et. al*. [2] proposes a backup warning policy for a hard disk of an engineering workstation or personal computer. A warning for backup operation is given at the elapsed time T_w since the last backup operation or the last recovery from hard disk failure. The main drawback of the proposed scheme is the non-responsive recovery due to unexpected hardware failures.

In reference [4], the authors propose a modification of the Hadoop Distributed File System (HDFS) called *DiskReduce* system. This enables asynchronous compression of the initially triplicate data and transforms it to a RAID – class redundancy overhead. This solution increases cluster capacity by a factor of three. The delay factor is a main concern in this technique and can affect its performance.

The References in [6 and 7] are used to show the benefits of applying the threshold sharing on network environments in terms of security and fault tolerance. The requirements to apply the threshold sharing on database systems are different from those in networking systems.

A data grid architecture concept for clinical image data backup is presented by Lui *et. al*. [15]. Clinical image data backup solutions are considered expensive and time consuming. Their study shows that the construction of a federation of picture archiving and communication system (PACS) archives serves as a cooperative backup archive, and that this solution can be effectively realized utilizing grid technology. In this design, only a small fraction of the PACS data archive resource is needed from each federated member. Furthermore, the massive overhead burden in system design development and operation can be mitigated by the public domain nature of most of the data grid technology.

Another important work that discusses data recovery in distributed database systems (DDBSs) is presented by Krishna *et. al.* [16]. The authors propose a Backup Commit (BC) protocol by including backup phases to the two-phase commit protocol. In reference [11], a one reverse hierarchical variable clustering on database accelerated algorithm is presented. The authors present a method that can both enhance the operation speed in large scale sparse database and save the storage space, simultaneously keep the inherent structure of the database. The main thought of this paper is by using a reverse

variable cluster method to produce a projection clustering database.

On the other hand, the security of database systems has been investigated by many researchers. The authors in [18] study security of the databases shared between many parties from a cryptographic perspective. Their focus was mainly targeting the confidentiality, privacy and integrity of the data. They have proposed a mixed cryptography database (MCDB) to encrypt databases over un-trusted networks in a mixed form using many keys owned by different parties. The work in reference [19] discusses privacy and security issues that are likely to affect data mining projects. Also, in reference [20], the authors propose an encryption-based multilevel model for database management systems. The proposed model is a combination of the Multilevel Relational (MLR) model and an encryption system. This encryption system encrypts each data in the tuple with different field-key according to a security class of the data element. Each field is decrypted individually by the field-key of which security class is higher than or equal to that of the encrypted field-key. Indeed, one database scheme which is very similar to this work is presented in reference [17]. However, it will be discussed later on after the proposed scheme is presented in order to have a better understanding for the difference between each one of them.

This proposed approach is different and utilizes the secret sharing scheme [6][7] to provide fault tolerance in addition to security by dividing the database into shares that are then distributed over an array of disks in such a way that would allow for full recovery when a sufficient (but not all) shares are available.

III. Proposed Approach

The main goal of this proposed recovery technique is to provide recovery when failures occur. This proposed technique for recovery uses a modified version of the threshold secret sharing scheme. The following discussion describes the basic idea of the secret threshold sharing scheme.

Secret sharing is a technique used for distributing a secret amongst a group of participants. Each participant is allocated a share of the secret. The secret can be reconstructed only when a sufficient number of shares are combined together; individual shares on their own are useless [3].

The Shamir sharing scheme [3] is used to implement this proposed technique for database recovery. In this scheme, any t out of n shares can be used to recover the secret, which is the database in this case. The system relies on the idea that you can fit a unique polynomial of degree $(t-1)$ to any set of t points that lie on the polynomial. For example, two points are needed to define a straight line, three points to define a quadratic curve, four points to define a cubic curve, and so on. In other words, it takes t points to define a polynomial of degree $t-1$. The scheme works as follows: First, a polynomial of degree t-1 is created with the original secret as the first coefficient while the remaining coefficients are picked at random. Next, n points on the curve are found and each is assigned to a different sharer. Only when at least t out of the n sharers disclose their points, there is sufficient information to fit a (t-1) degree polynomial to them. The first coefficient of the polynomial is the secret (the original database as in this case).

The Shamir (k, n) threshold scheme is based on the Lagrange interpolation for polynomials. The polynomial function is as shown in equation (1), where p is a prime number, coefficients a_1 , … , a_{k-1} are unknown elements over a finite field Z_p, and a_0 = M is the original data.

$$f(x) = (a_{k-1}x^{k-1} + \ldots + a_2x^2 + a_1x + a_0) \bmod p \quad (1)$$

Using Lagrange linear interpolation the polynomial function can be represented as follows:

$$f(x)=\sum_{j=1}^{k} y_{ij} \prod_{1\le s\le k, s\ne j} \frac{x - x_{is}}{x_{ij} - x_{is}} \quad (2)$$

In the following section we describe the architecture and functionality of this proposed algorithm.

A. Database Protection Using Threshold Secret Sharing (Architecture of the Proposed Work)

This subsection presents the high-level system architecture for the proposed recovery technique. Figure 1 depicts a simplified architecture of the system. The basic operation of this technique can be summarized in the following points:

1. The database information that is passed into or from the storage device has to go through an interface component, i.e., the Database Controller. The Database Controller component shown in Figure 1 is responsible for distributing and reconstructing the data.
2. The new total database size after applying the threshold sharing scheme is equal to:

   Total Database Size = n/k × *Original Database Size*

   (3)

   Where, n = Total number of shares

   k = Number of shares required to reconstruct the original data.
3. The proposed technique can provide data recovery for a single failure if (n–k=1), or multiple failures if (n-k>1).

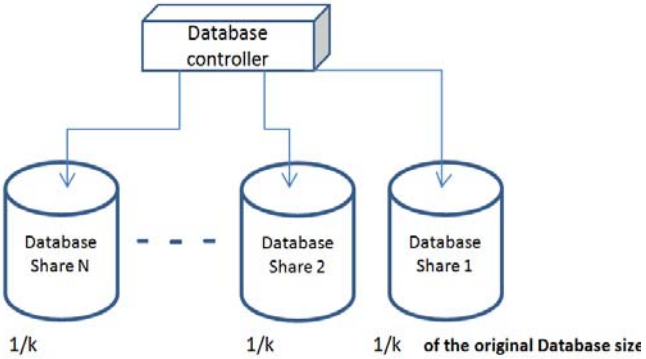


Figure.1 Database partitioning using the Threshold Sharing Scheme

From the above discussion, it is clear that the Database Controller is the basic component of this system. To illustrate the general function of this component, the e-mail storage system is taken as an example. The database that is to be protected is the users' mail boxes. Now, when the e-mail server receives new e-mails or updates e-mails in the system, it has to perform this task by distributing the data to sub-database shares through the Database Controller that runs the threshold sharing scheme

Algorithm 1: [Distributing the Original Database into Sub-storage Partitions]
Input: a binary file *S* (e.g., object, executable EXE, etc) obtained from the original Database
Output: *n* binary files distributed into sub-storage databases, depending on the (*k*,*n*) TSSS.

Step1: *Initialization*
- Let a binary file of size S be fed to the TSSS, determine the size of *S* in bytes.
- Select the TSSS level, set *k* and *n* values.

Step 2: *Processing TSSS*
- *Break* the binary file into blocks of fixed lengths B_1, B_2, …, B_i, where $0 \le i \le U$, and U is the total number of blocks in a binary file S.
- Block size should be divisible over *k* with no remainder, i.e., $(Bi/k) \neq 0$.
- *Padding*, in the last block in *S*, padding is required if number of bytes in the block is not divisible by *k*.
- In each block, *all* bytes are fed *to all* coefficients of the polynomial function $f(x)= a_{k-1}x^{k-1} + \ldots + a_2x^2 + a_1x + a_0$.

Where
a_0 = byte 1 of B_i
a_1 = byte 2 of B_i
a_{k-1} = byte *c* of B_i, where *i* is byte index in a block B_i

- *Select* the irreducible polynomial for $GF(2^8)$ finite field to be $m(x) = x^8 + x^4 + x^3 + x + 1$.
- Calculate *n* values for function f(x), where x=1,2,3,…*n*.
- Save and distribute the calculated *n* values to *n* output binary files in sub-storage databases.

Step 3: *Repetition 1 (Inside block repetition)*
Repeat Step 2 for all bytes in B_i

Step 4: *Repetition 2 (Inside binary file repetition)*

Repeat Step 2 and 3 for all blocks in the binary file S.

Step 5: *Finish* the TSSS processing or *wait* for another binary files to be processed.

B. The Design Model of This Implementation

The threshold sharing scheme presented in reference [7] will be used to implement the database partitioning. In order to reduce the redundancy overhead, the requirement of the random selection of coefficients in equation (1) above will be changed to use actual values obtained from the original database.

Figure 2 below shows an example of how the distribution process applies a (2, 3) Threshold Secret Sharing scheme into a binary file. The binary file is first divided into m blocks B_1, B_2, ... , B_m where each block size L is a multiple of k bytes. The effect of block size L on file processing is shown later in Figures 3 and 4. From Equation (1), it can be easily seen that there are two coefficients, *a*0, and *a*1, for a (2, 3) scheme where k = 2. Each block is therefore divided in *k* equal parts and these coefficients are assigned values from the block (unlike original TSSS scheme where *a*1 is assigned a random value).

Let us take another example with *n*>*k*. The p value is equal to 31. Let the threshold be *k*=3, and the input to the threshold sharing system are the following groups of data: *a*1= x7, *a*2=x19, and *a*3=21 in Z_{31}. $f(x) = 7+19x+21x^2$. In this case, the system can generate as many shares as the system requires. In other words, n=3 if no redundancy is needed, or *n*= 4 if recovery from one failure is required, and so on. Let us say that one redundancy is required, then the following shares can be produced:

(x=1, f(1))= (1,16)
(x=3, f(3))= (3,5)
(x=5, f(5))= (5,7)
(x=7, f(7))= (7,22)

From any three values of the above share, the polynomial coefficients *a*1, *a*2 and *a*3 can be reconstructed using Lagrange interpolation.

Lemma 1: The following conditions should be fulfilled between the coefficients and the polynomial variables for the multiplication inverse:

For each $w \in Z_p$, $w \neq 0$, there exist $z \in Z_p$ such that $(w \times x) \equiv 1 \pmod p$. We further observed that any integer in Z_p has a multiplicative inverse if and only if that integer is relatively prime to *p* [14]. The finite field of order p^n is generally written $GF(p^n)$; Either of the following case can be considered. The first one is when n = 1, we have the finite field GF(p); this finite field has a different structure than that for finite fields with n > 1 and it is the one which is going to be applied in our implementation studied in this section. We look at finite fields of the form $GF(2^n)$.

Suppose it is needed to define a conventional coding algorithm that operates on data 8 bits at a time and also it is needed to perform division. With 8 bits, it can represent integers in the range 0 through 255. However, 256 is not a prime number, so that if arithmetic is performed in Z_{256} (arithmetic modulo 256), this set of integers will not be a field. The closest prime number less than 256 is 251. Thus, the set Z_{251}, using arithmetic modulo 251, is a field. However, in this case the 8-bit patterns representing the integers 251 through 255 would not be used, resulting in inefficient use of storage. The same argument applies when selecting 257 as a prime number where the waste

will be higher. An equivalent technique for defining a finite field of the form GF(2^n) using the same irreducible polynomial, is sometimes more convenient. Indeed, with the irreducible polynomial $G(x) = x^8 + x^4 + x^3 + x + 1$, this methodology can perform Multiplication and Addition operations on equations (1 and 2).

Having defined the finite field requirements for this approach, the following algorithms (*Algorithms 1 and 2*) illustrate the steps of processing threshold sharing in a database in order to create, distribute the sub-storage partitions and the reconstruction of the original database:

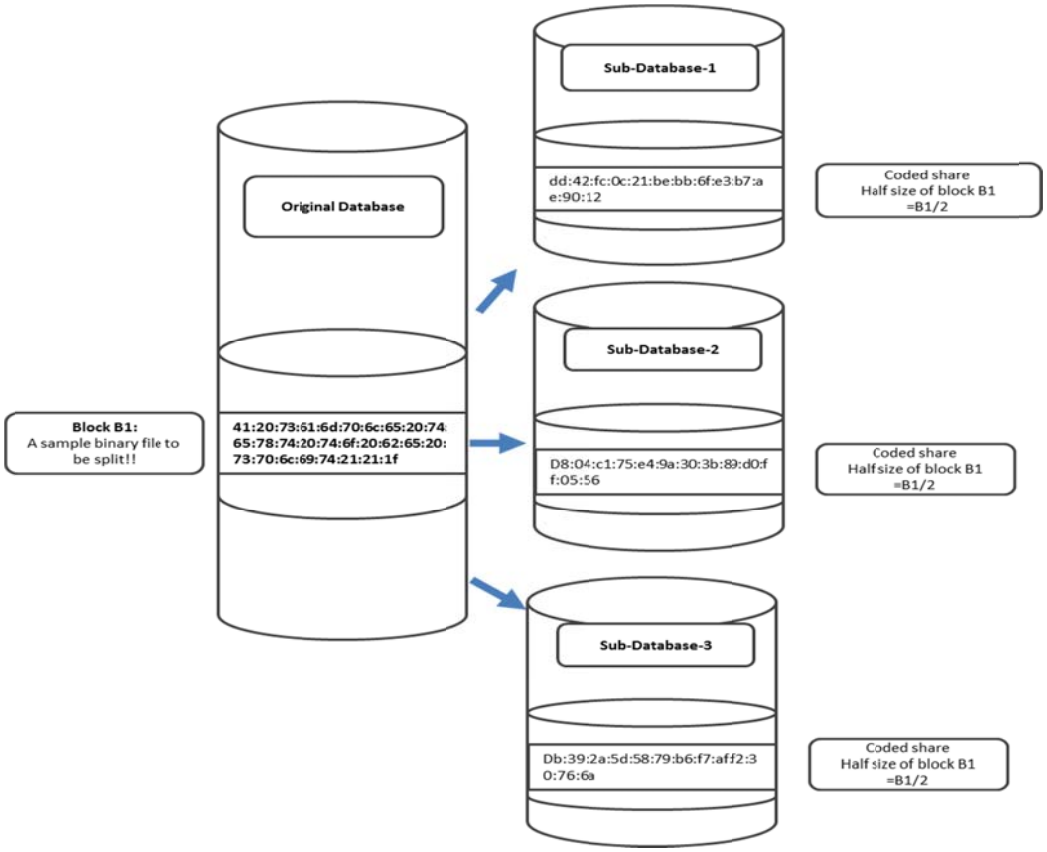


Figure.2 Example for a distribution process on a block of data

C. Security Enhancement for the Database System

The application of TSSS on the database system can provide four main security features which are:

Confidentiality of the database

Now, from a security point of view, data confidentiality can be the only security level required in a database system, and therefore, this requirement can be achieved using a (*k, k*) TSSS. This means that no additional sub-storage share(s) is/are required. The data confidentiality is basically the main goal of the original threshold secret sharing algorithm. The shares generated by the distributor process carry an encoded form of the original database. The confidentiality of data is actually inherited from the original TSSS algorithm.

Detection of database modification:

To support data integrity it is required to apply a (*k*, *n*) TSSS scheme where n > k. This means extra or redundant shares are needed. The impact of adding more shares on the network resource utilization is shown in equation (3). The detection of modified share(s) can simply be obtained by comparing values reconstructed from different groups of shares.

Identification of modified shares

To support identification of modified shares, the following requirement should be available in a (*k*, *n*) TSSS scheme, which is $n > k + 1$.

Availability

This work focuses on Denial of Services (DoS) as an example of availability attacks on database systems. A denial of service attack is an incident in which a user or organization is deprived of the services of a resource they would normally expect to have. Although a DoS attack does not usually result in the theft of information or other security losses, it can cost the target person or company a great deal of time and money. Using a (*k*, *n*) TSSS where n > k, the service continues operational if (*n-k*) database shares are under DoS attacks, since *k* database shares are enough to reconstruct the original database. Indeed, this is again inherited from the basic TSSS model.

Algorithm 2: [Recovering the Original Database from Sub-storage Partitions]

Input: binary files shares from sub-storage databases
Output: a binary file of the original database

Step 1: *Select* any k out of n binary files share.

Step 2: *Perform Lagrange interpolation* as in equation (2) for any k sub-storage blocks.

- *Select* the irreducible polynomial for $GF(2^8)$ finite field to be $m(x) = x^8 + x^4 + x^3 + x + 1$.
- *Recover* the coefficients of polynomial $f(x) = a_{k-1}x^{k-1} + \ldots + a_2x^2 + a_1x + a_0$. Each coefficient represents a byte in the original block Bi of the S binary file.

Step 3: *Reconstruction*
Reconstruct the original block Bi from blocks B_{sub} in each sub-storage database, where B_{sub} is the block size in the binary file share.
Step 4: *Repetition*
Repeat Steps 2 and 3 until all blocks from sub-storage databases have been reconstructed and finally the original binary file is recovered.

IV. PERFORMANCE EVALUATION

The measurements were performed on 3 GHz Intel Core 2 Duo processor with 4 GB memory running under Linux operating system to measure the time taken by this TSSS scheme to process variable file sizes for the distribution and reconstruction processes.

A. Write and Read Operations

In this section we evaluate the performance of the *write* and *read* operations. These operations are performed on the disk drives through the Database Controller and buffering is performed using a cache memory installed in the Database Controller. The required buffering time for the distribution and reconstruction processes is modeled accordingly. Figure 3 below shows the distribution processing time that is measured for variable database sizes.

It is worth to note that the processing times for the database shares reconstruction are found to be close to those obtained for the distribution process. Therefore, the figures 3 and 4 do not specify the processes.

The results obtained from Figure 3 above show that the processing times required to distribute the original data using the (*k,n*) level in the TSSS application increase as the data size increases. This increase is somewhat linear and thus does not favor larger sizes over smaller ones, nor smaller sizes over larger ones. In the figure, we have kept the value of N as variable. In other words, the N values used in the testing were checked for different n≥k (e.g., level (5,5), (5,6), (5,7), level (4,4), (4,5), (4,6), and level (3,3), (3,4), (3,5)), nevertheless in all cases the processing time results for each (k,n) recovery level were not affected by the N values.

Verifiable Secret Redistribution (VSR) model is another software recovery technique that was proposed by Wong [10] and uses threshold secret sharing in software data recovery. VSR distributes data storage in multiple servers to preserve it for long times. A comparison is made between the processing times of this technique and the VSR model, the results are presented in Figure 4. It is worth to note that this approach is a typical implementation of the original threshold sharing scheme. In other words, with reference to equation 1, coefficients a_1 , … , a_{k-1} are unknown element which are given random values and a_0 is the original database file to be processed. This exact implementation of the original Shamir threshold sharing scheme makes the recovery redundancy cost in terms of storage size *n* times higher compared to this approach which uses a threshold sharing scheme. The MIRACL package for arithmetic integer precision is used in the implementation of VSR model while this implementation uses the Multiple Precision arbitrary sized integers (MPZ) functions under GNU Multiple Precision (GMP) Arithmetic Library.

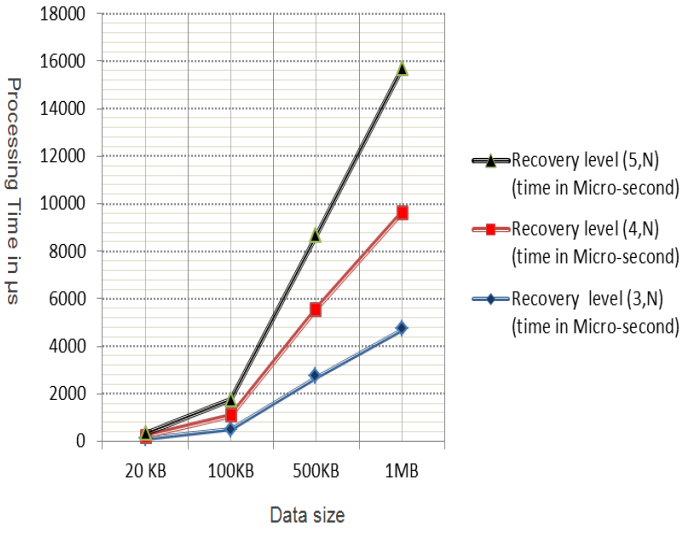
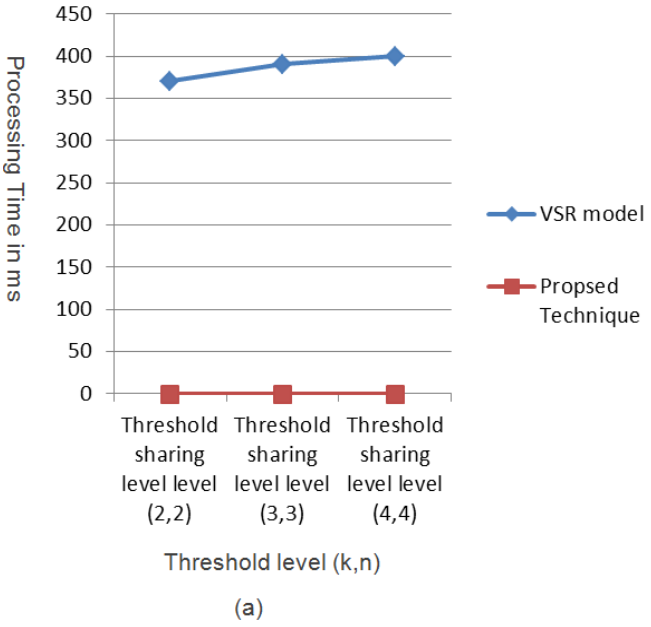


Figure.3 Database distribution processing times



(a)

Figure.4 (a) Graphical representation, Data shares distribution processing times in the VSR model compared to t proposed technique using the same testing files sizes.

Threshold Sharing Level	VSR Model	Proposed Technique
(2,2) Level	370 ms	20 µs
(3,3) Level	390 ms	55 µs
(4,4) Level	400 ms	90 µs

(b)

Figure.4 (b) Tabular representation, Data shares distribution processing times in the VSR model compared to t proposed technique using the same testing files sizes.

B. Failure Recovery Comparison with POTSHARDS Archival System

The storage archival system proposed by Storer *et. al.* [17] uses secret sharing to provide archiving security by splitting and spreading the resulting shares across separately managed devices, this developed system is named POTSHARDS. There is a major difference between this proposed work and the POTSHARDS system which is the storage overhead requirement. To illustrate more, to provide data integrity in POTSHARDS, a two-way XOR split followed by a (2,3) secret split increases storage requirements by a factor of six. On the other hand, this proposed solution requires only one-third of the original storage size.

This is a very significant gain in this approach compared to the POTSHARDS. This modification to the secret threshold sharing original model in reference [3] enabled us to provide this reduction. The POTSHARDS system implemented the original Shamir secret sharing technique [3] as is, and the result will be having a considerable overhead which is a result from the secret sharing itself, and that is why it is required to modify the Shamir secret sharing model first and then it can be used in database storage system.

RAID is implemented at a lower level either in hardware or software. To compare with RAID, this proposed approach is integrated into a database system at a file level for a database. Using MySQL as an example, and depending on the implementation model used, the database in this implementation consists of a variable number of large or small binary files. RAID on the other hand; consists of small number of large binary files.

In order to overcome a single failure of one of the sub-database shares, the database controller has to apply the n-k=1 condition. This condition used in this technique is comparable to a RAID 5 level. The parity XOR is used in RAID 5 to calculate the parity drive. On the other hand, this technique implements the Lagrange interpolation for polynomials.

To calculate the complexity of this scheme, it is needed to identify the elements of the Database Controller. In the following discussion the complexity of the distribution process is to be discussed.

The complexity of the distribution process is deduced from Figure 2; it is expressed in terms of the original data size, the size of the blocks to be used, and the number of database shares [7]. More precisely, if M is the size of the original database, B is the size of the blocks resulting from the division of the original data, and N is the number of database shares, then the complexity of the distribution process is:

$$\text{Distribution process complexity} = O(\frac{M}{B} \times N) \quad (5)$$

In conclusion, the distribution process complexity of the XOR parity in RAID 5 and the threshold sharing are theoretically the same. On the other hand, to calculate the complexity of the reconstruction process of the threshold sharing scheme, this requires the identification of the number of database shares needed, the number of blocks used, and the complexity of the Lagrange linear interpolation. So, if a is the number of required database shares and b is the number of blocks used, then the complexity of the reconstruction process is

$$O(b \times a^3) \quad (6)$$

The RAID 5 reconstruction process has the same complexity as the distribution process. The reconstruction process becomes more complex as the RAID level increases, i.e., RAID-6. On the other hand, this algorithm has the same complexity degree no matter what the value of the threshold level is [9].

C. Failure Detection in Database

Failure detection of data written onto or read from hard disks is a crucial requirement that has to be available in any recovery technique. The detection of failure in RAID technology is available in RAID levels 5 and higher.

In this technique, the detection of database failure can be achieved by threshold sharing when n=k+1.

The detection of failed sub-database share(s) can simply be obtained by comparing values reconstructed from different groups of database shares as shown in Figure 5. The original database (ODB) is divided into three sub-database (SDB) shares *r1*, *r2*, *r3* using a (2,3) threshold sharing scheme. The reconstruction process requires a group of at least two SDB shares to be able to reconstruct the ODB. The number of groups which contains different combinations of shares is given by:

$$G_{(k,n)SDB} = \binom{n}{k} \quad where\ n > k \tag{7}$$

For $n = k+1$, the number of groups $G_{(k,\,k+1)TSSS}$ is equal to:

$$G_{(k,k+1)SDB} = \binom{k+1}{k} = k+1 \tag{8}$$

In Figure 5, an example of a database recovery (R_{DB}) using a (2,3) threshold sharing scheme is presented. Based on equation (8) the total number of groups obtained is 3. In Figure 5-(a), if a failure occurs at SDB_3, then only one group reconstructs the original data which represents the true value of the original database (ODB), where the other two reconstructed database values have different "invalid" values. Also, the failure of two or three SDB shares will result in different "invalid" reconstructed database values as shown in Figures 5-(b). In conclusion, the (k, $k+1$) threshold sharing scheme can successfully provide detection of data failure.

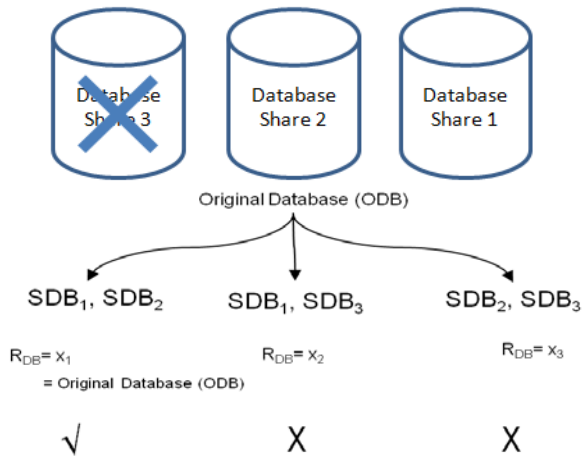


Figure.5 – (a) An example of a (2,3) threshold sharing scheme used to detect database failures with single sub-database failure

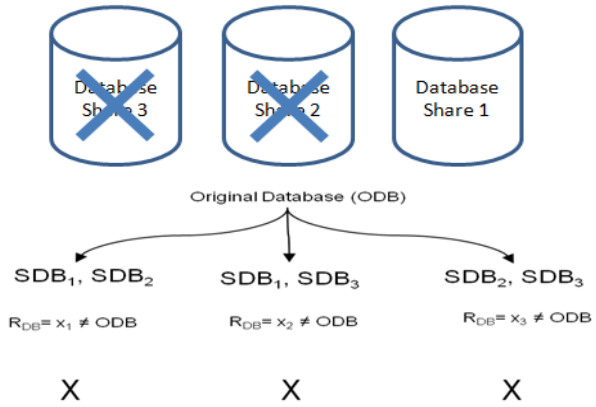


Figure.5 – (b) An example of a (2,3) threshold sharing scheme used to detect database failures with multiple sub-database failure detection

D. Database Repair (Correction)

In order to repair/correct a failed database, the location of the failed SDB share should be found. It is worth to note that using a (k, $k+1$) sharing scheme; there will be always one correct reconstructed original database with only one sub-database failure, or no correct reconstructed values if more than one sub-database share have failed. This result is shown in Figure 5. The explanation of this result comes from the fact that if there is one failed database share, then only one group is able to reconstruct the original database from the non-failed shares. This result can be formalized by the following equation:

$$G_{(k,k)SDB} = \binom{k_{non-failed-shares}}{k_{non-failed-shares}} = 1 \tag{9}$$

On the other side, if there are more than one failed sub- database shares, then there will be no group that can contain $k_{\text{non failed shares}}$.

From the above discussion, the location of the failed sub-database share cannot be determined as there are at maximum one group which is able to reconstruct the original database given that one failed database share occurred.

Therefore, in order to be able to repair the failed sub-database share, it is needed first to locate it and differentiate it from the other valid shares. To do so, a higher (k, n) level is required, i.e., (k, k+2) level can be an appropriate choice for a single failure of sub-database shares. It is worth noting that this repair option requires the database system to allocate more bandwidth space size.

Furthermore, in the case of power failure of one of the sub database shares, the detection of the failure can be pointed to the element share that has a power failure. Also, it is easy to determine which share element has failed and therefore, no higher (k, n) level is required. However, to make this work more comprehensive, and if resources are available, it is preferable to build this proposed recovery system with higher (k, n) levels.

E. A Future Work Application

For future work, an extension to this technique is proposed to allow its implementation over computer networks with high speed and bandwidth capabilities. The

basic idea is to distribute the sub-database shares over separated network locations. In other words, this approach spreads the database shares among physically separated locations in order to protect and recover from catastrophic failures such as fires and earthquakes [12]. This is indeed a vital component of business continuity.

An example of a preliminary network implementation of this extension is shown in Figure 6. The sub-database shares are distributed at the edge routers ER3, ER4, and ER5. The edge routers belong to the same network domain. The implementation of this approach should take into consideration the following points:

a. The Database Controller application is installed in one of the edge routers, e.g. ER1 as shown in Figure 6.
b. Each sub-database share should create a path with the Database Controller router, e.g. a path between ER3 and ER1 {ER3 → R8 → R1 → ER1}.
c. It is preferable that the paths created between each database share and the Database Controller be maximally disjoint. That is, the paths between the sub-database shares and Database Controller include a minimum number of shared nodes between them.
d. The Database Controller router is responsible for the process of maintaining the paths with sub-database shares. This process should take into account variable path lengths.

To sum up, the architecture of this future network-based database protection technique requires the threshold sharing application to be installed on a server (i.e. the server here refers to the Database Controller). Therefore, any information that has to get into or from the database should pass through this server. One drawback of such approach is the single point of failure. However, this problem can be overcome by adding another server to act as a backup server when the primary one fails. This extra server can also support load balancing.

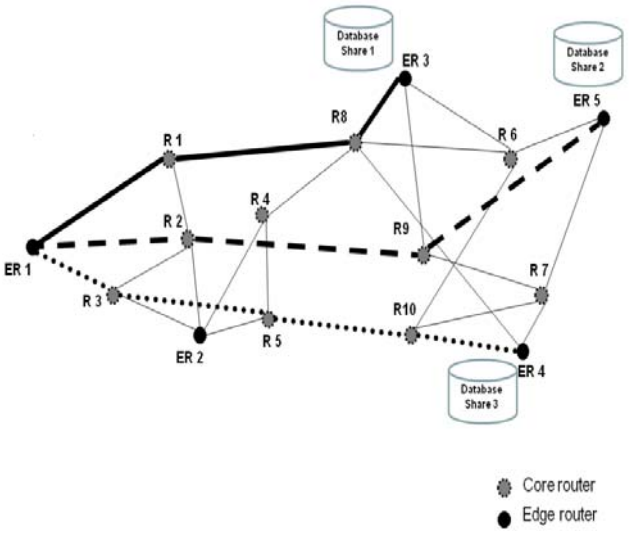


Figure.6 A network topology example to illustrate the distribution of database shares.

The selection of sub-database locations in the network should take into consideration that the paths lengths can be variable and therefore the Database Controller server has to buffer the data received from different sub-databases in order to be able to reconstruct the original database when all shares are received. In other words, it is only required to receive *k* out of *n* sub-database shares in order to reconstruct the original database. Therefore, in case one sub-database storage location failed or the path between this storage location and the database controller server is broken, this will not affect the reconstruction process. Future work on the application of this work on networking will be carried out to confirm its feasibility and performance. Our solution can also be applied in Multicast networks to provide recovery and security [7,and 13].

V. CONCLUSION

In this paper, a new technique is presented to protect database systems from failures due to data corruption, power failures, and/or network failures. This proposed technique is based on the threshold sharing scheme. We demonstrate the effectiveness and feasibility of this proposed technique and have demonstrated the modifications on the original threshold sharing scheme by explaining the distribution and reconstruction processes. Moreover, these results highlight the fact that this solution imposes insignificant time delays and data overhead in order to distribute and recover the data, making it in all a very attractive solution. Also a comparison has been made between this approach and the RAID technology and POTSHARDS archival system.

The security benefits of this proposed technique have also been discussed for the purpose of providing confidentiality, detection and identification of modified database, and availability. Furthermore, a future work for a network-based application of this technique is presented.

REFERENCES

[1] S. Nakamura, C. Qian, S. Fukumoto, and T. Nakagawa, " Optimal Backup Policy for a Database System with Incremental and Full Backups", Mathematical and Computer Modelling journal, Elsevier, Vol. 38, 2003, pp. 1373-1379.
[2] H. Kawai, and H. Sandoh, " An Efficient Backup Warning Policy for a Hard Disk", Computers and Mathematics with Applications, Vol 46, 2003, pp. 1055 - 1063.
[3] A. Shamir, “How to share a secret,” Communications of ACM, vol. 24, Nov. 1979.
[4] B. Fan, W. Tantisiriroj, L. Xiao, and G. Gibson, “DiskReduce: RAID for data-intensive scalable computing”, In *Proceedings of the 4th Annual Workshop on Petascale Data Storage* , November 14, 2009), PDSW '09. ACM, New York, NY.
[5] P. Ammann, S. Jajodia, and P. Liu, “ Recovery from Malicious Transactions”, IEEE Transactions on Knowledge and Data Engineering, Vol. 14, No. 5, Spet. 2002.
[6] S. Alouneh, A. Agarwal, and A. En-nouaary, “A Novel Path Protection Scheme for MPLS Networks Using Multi-path Routing”, Journal of Computer and

Telecommunications Networking, Elsevier, Vol. 53, Issue 9, June 2009, pp. 1530-1545.
[7] S. Alouneh, A. En-nouaary, and A. Agarwal, “MPLS Security: An approach for Unicast and Multicast Environments”, Annals of Telecommunication, Springer, Vol. 64, Issue 5, 2009, pp. 391-400.
[8] M. Iwaki, K. Toraichi, R. Ishii, “ Fast Polynomial Interpolation for Remez exchange Method”, IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 411-414, 1993.
[9] P. Anvin, “The mathematics of RAID-6. H. Peter Anvin”, Article, Kernel.org. 2009.
[10] T. Wong, “Decentralized Recovery for Survivable Storage Systems”, Ph.D Thesis, Carnegie Mellon University, 2004.
[11] W. Lu, P. Zhou, and X. Liao, “Reverse hierarchical variable clustering on database accelerated algorithm”, System Science and Engineering (ICSSE), 2011 International Conference on , vol., no., pp.251-254, 8-10 June 2011.
[12] Y. Ping, K. Bo, L. Jinping, and L. Mengxia, “Remote disaster recovery system architecture based on database replication technology”, Computer and Communication Technologies in Agriculture Engineering (CCTAE), 2010 International Conference On , vol.1, no., pp.254-257, 12-13 June 2010.
[13] J. Cui, M. Faloutsos, M. Gerla, “ An Architecture for Scalable, Efficient, and Fast Fault-Tolerant Multicast Provisioning”, IEEE Networks, pp. 26-34, March/April 2004.
[14] William Stallings, “Cryptography and Network Security Principles and Practices”, Fourth Edition, Prentice Hall.
[15] B. Lui, M. Zhou, and J. Document, “ Utilizing data grid architecture for the backup and recovery of clinical image data”, Journal of Computerized Medical Imaging and Graphics, ELesevier, Vol. (29), pp. 95-102, 2005.
[16] P. Krishna, and M. Kitsuregawa, “ Reducing the blocking in two-phase commit protocol emplying backup sites”, Proceedings of Third IFCIS Conference on Cooperative Information Systems, 1998.
[17] M. Storer, K. Greenman, and Ethan Miller, “POTSHARDS: Secure Long-Term Storage Without Encryption”, 2007 USENIX Annual Technical Conference, USA, pp. 143-156, 2007.
[18] H. Kadhem, T. Amagasa, and H. Kitagawa, "A Novel Framework for Database Security Based on Mixed Cryptography," Internet and Web Applications and Services, 2009. ICIW '09. Fourth International Conference on , vol., no., pp.163-170, 24-28 May 2009.
[19] K. Pathak, N. Chaudhari, A. Tiwari, “Privacy-Preserving Data Sharing Using Data Reconstruction Based Approach”, IJCA Special Issue on Communication Security comnetcs(1):64-68, March 2012. Published by Foundation of Computer Science, New York, USA.
[20] A. Sallam, E. El-Rabaie, and O. Faragallah, “Encryption-based multilevel model for DBMS”, Computers & Security, Elsevier, Available online 22 February 2012, ISSN 0167-4048.