Privacy Preserving using Spider Security Pattern Generation based on ORB Web Construction

V.Sravan Kumar, Rashmi Agarwal

ABSTRACT--- ***The aim of privacy preserving data mining algorithm is to extract relevant knowledge from large amount of data while protecting at the same time sensitive information. A number of algorithmic techniques have been design for privacy preserving data mining. Base on the orb web construction step in spider, the algorithm is proposed to construct the artificial orb web. In this work the uniform orb web is selected. The proposed algorithm consist of four major processes which are the construction of initial radial lines other radical line and frame initial spiral line and other spiral line. The centre position and the number of radial lines can be changed as per the requirements. Every spider has some gland and each gland is able to produce spider silk.***

Keyword: Privacy Preserving, Spider orb web, Spider Silk, Radial Lines.

I. INTRODUCTION

A powerful tool in computing a wide range of functions with computational security is homomorphic encryption. With homomorphic encryption, we can avoid the bitwise encryption from the scrambled circuits described. Homomorphic encryption schemes are a special class of public key encryption schemes. The first homomorphic cryptosystem, called the Goldwasser-Micali (GM) cryptosystem, was proposed in 1984[11]. Due to its prohibitive message expansion during encryption (i.e. each bit of plaintext is encrypted as a cipher text of at least 1024 bits), it is not practical for data mining applications. The natural extension of the GM cryptosystem is the Benaloh cryptosystem [21], which allows the encryption of larger block sizes at a time. Although the message expansion is not as bad as in the GM cryptosystem, it is still not suitable for data mining applications. Furthermore, the fact that the decryption is based on exhaustive search over all possible plain-texts also makes the Benaloh cryptosystem unpractical for privacy preserving data mining. A more recent scheme is the Paillier cryptosystem [18], which avoids many of the drawbacks of the earlier homomorphic cryptosystems. The Paillier cryptosystem provides fast encryption and decryption algorithms, and it encrypts 1024-bit messages in cipher texts of at least 2048-bits, which is reasonable if we work with large plaintexts. Homomorphic encryption enables us to compute certain functions more efficiently compared to scrambled circuits. The authors of [9] use homomorphic encryption for computing secure scalar products used in privacy preserving data mining. The protocol is shown in Fig. 1, where two players, A and B, compute the scalar product of vectors *v* = (*v*1, . . . , *vd*), and *w* = (*w*1, . . . , *wd*), such that only B learns the scalar product, and A learns nothing at all.

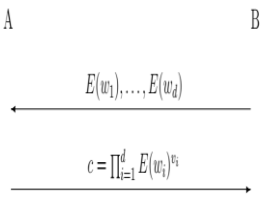


Figure 1: Computationally secure scalar product protocol (*D*(*c*) = *v* · *w*)

II. RELATED WORK

Privacy preserving joint Gaussian noise added privacy preserving scheme

EDEX is processed in five stages

(1) extract all super entities in the source schema
(2) add Gaussian noise with all attributes
(3) prune redundant entities
(4) select the best host relations for these entities in target
(5) move the pruned super entities to their proper host tables

Step 1 (Super Entity Generation). The initial stage for data exchange in EDEX is extracting all super entities as they contain whole information related to source entities irrespective of classification. The edge between the nodes v_1 and v_2 in the schema graph is a foreign key from a v_1 column to the primary key in the v_2 column. Each foreign key of a tuple references at most one tuple of the referenced table (where more than one foreign key references the same table, the tree includes more than one edge with different labels between the nodes).

Manuscript published on 28 February 2019.
* Correspondence Author (s)
V.Sravan Kumar*, Department of Computer Science and Engineering, Madhav University Rajastan, India. (E-mail: sravanresearch@gmail.com)
Rashmi Agarwal, Department of Computer Science and Engineering, Madhav University Rajastan, India. (E-mail: rashmiagarwalcse@gmail.com)

A super entity with respect to tuple t is a flat structure described as an over view of all ancestors of t. RAT(r) *(*Relation Ancestor Tree) is a structure to depict this view that can be built with respect to rows (i.e.) tuples of relation r. Initially, the node r is selected as a root to build *RAT(r)*. Next, all the outgoing edges of 'r' and their respective nodes of 'R' are joined to 'r' using the schema graph. The outgoing edges of each node r_i in R and their respective nodes are added to r_i if they are not present already in the *RAT(r)*. This addition continues until there is no edge to sum or summing edges results in a loop in *RAT(r)*. As soon as Relation Ancestor Trees are extracted from the all relations in the schema, super entities can be find from the view statements generated using post-order traversing these trees. The leaves are connected with parent nodes in each step. The outcome of this traverse is a representation of jointed nodes in a nested view statement. A Relation Ancestor Tree is constructed in post-order manner where each stage forms the connection between the parent and child.

Step 2 (Pruning Redundant Information). The group of super entities must be cut down to reduce the repetitive information. So here we now introduce the term of distinct super entity, a distinct super entity is an entity that possess at least one unique property that does not present in other instance of super entities. A pruner algorithm is used to trace out the list of distinct super entities and to check if all the attributes of a super entity (the set of ‹property, value› pairs specifying that super entity) present in at least one other super entity.

Step 3 (adding Gaussian noise). Gaussian noise is a statistical noise has a PDF **(**probability density function) equal to the normal distribution also termed as Gaussian distribution. It can also be explained as values that the noise can take on are Gaussian-distributed.

The PDF (probability density function) p of a Gaussian random variable z is given by:

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$

{\displaystyle p_{G}(z)={\frac {1}{\sigma {\sqrt {2\pi }}}}e^{-{\frac {(z-\mu)^{2}}{2\sigma ^{2}}}}} Where {\displaystyle z}z represents the grey level, {\displaystyle \mu }μ the mean value and {\displaystyle \sigma }σ the standard deviation.

The white Gaussian noise is a special case where the values at any instance of time are statistically independent and identically distributed (and hence uncorrelated). Gaussian noise is an additive white noise used in producing the additive white Gaussian noise in communication channel modeling and testing.

A wideband Gaussian noise from natural sources like shot noise, thermal vibrations of atoms in conductors (referred to as thermal noise or Johnson-Nyquist noise), from celestial sources such as the Sun and black body radiation from the earth and other warm objects can affect the communication channels in telecommunication and computer networking.

Step 4 (Host Relation Selection). Several issues have to be considered on selecting the target host. First issue is the presence of same concept with different representations and with different properties in both source and target. So property correspondence in the form of *‹p1, p2›* depicting the correspondence between the property *p1* and *p2* in the source and target is used to connect the source and target. Every correspondence semantically shows the relation between the attributes in the target and source. The proposed approach selects the best host with respect to source entities using value correspondences regardless of schema mapping expression. We considered some conditions for picking up the suitable host for source entities:

(1) Completeness: It means residing host must have the capability to recover the properties of source entities in the target as soon as possible.

(2) Non-redundancy: It means there is no repetitive information is transferred to the target. We developed the host selection algorithm (Algorithm II) to satisfy this condition. We use the target schema in Figure 6 and the following value correspondences *{name ↔ stName, program ↔ prog, dName ↔ dpt, supervisor ↔ supervisor, course ↔ courseName, regDate ↔ date}* between this schema and source schema to explain the host selection algorithm.

We considered the relation ancestor trees in the target as structures that can reside super entities of the source to select the suitable hosts for source entities. Next, the RAT correspondence to each relation in the target must be traced out. Then, we need to find the structure that can correctly reside super entities of source. It shows the RATs constructed for each relation in the target. We assume existence of a unique property name for each property (as each property can be named using the triple (*dbName, tableName, proprtyName*). We create the hash table with the value correspondence where the correspondence property of each correspondence *‹p1, p2›*, of a given property is accessible. The best RAT (Relation Ancestor Tree) for residing the given super entity is the one having maximum number of properties (class-level properties) matching with the super entity properties. Among those *RATs* with maximum number of matching properties, a *RAT* with minimum number of total properties is selected as this *RAT* holds minimum number of unrelated properties

Step 5 (Entity Residing). The last step in EDEX is residing the super entities structured out from the source in host RATs. Even though the super entities repetition in the other entities are removed in second step by pruner algorithm. Still, there is some left super entities containing the repeated information about the same entity. For example, the super entities *e1* and *e2* contains data regarding the same department. In data exchange, such data must denoted by the same entity in the target to prevent entity redundancy. This problem is addressed here by using the target egds to encode primary key constraints in the target. These constraints check the primary keys for avoiding the insertion of the same entities in different location with different identifications. If a request to insert in target relation is made, then algorithm checks the redundancy of the unique properties to be inserted.

If it confirms the redundancy, the insertion is declined or the insertion is performed. The one significant issue to be considered in residing of super entities is that data of the ancestor has to be inserted previous to the data about the descendants, as each descendant has its own primary property referring the primary key of their parent. Mainly, the structure of the host RAT gives proper information about the order of insertion. Each host RAT has a post-order traversal to ensure about the insertion of parent data before child data insertion. The details of generating insertion statements to reside a super entity in a host RAT is shown in Algorithm III. As this algorithm is implemented in our EDEX prototype, the insertion statement produces by the algorithm can be changed in to SQL insertion statement easily. The host RAT is traced in a post order manner and the nested structure is created for insertion. For example, create a host RAT for the relation *Reg*, the following expression were created depicting the insertion order. *ex1:Reg(student (ST: stName, prog, dpt, supervisor), cName (Course: courseName, credit), date)*. This structure depicts the insertion order of given properties of a super entity. Thus we start from the greatest ancestors to create the insertion statements. In our example, *ex1* prescribes three insertion statements with the order of ST, Course and Reg. First, two set of properties P1 = (*stName, prog, dpt, supervisor*) and P2 = (*courseName, credit*) are inserted as two nested set of properties. For example, for a super entity *e7* (*e7: {(sName, S1), (name, S1), (program, P1), (dep, D1),(dName, D1),(building, B1), ((supervisor, prof1), (pName, Prof1), (degree, deg1), (profDep, D1), (course, C1), (regDate, dt1)}*) with *Reg* as a target host, the algorithm first checks if there is a common property between properties of *e7* and P1. In this case *{stName, prog, dpt, supervisor}* are selected as common properties. The value correspondences are considered in finding the common properties (e.g., *dName* in source corresponds to *dpt* in target). Then, with respect to the primary key of the corresponding target relation *ST* (i.e., *stName*), the ST relation is checked for the data about student in existing tuples, if not the tuple containing these properties will be inserted and the generated primary key will be returned as reference. If already the tuple exist, then *stName* will be returned as reference. In the same way, for the second nested set of properties (*Course: courseName, credit*), *courseName* is identified as common property between *e7* and *ex1*. Then, *Course* table is checked to find if information regarding *cName=C1* is already inserted to *Course*. Finally, since there is no other nested statement, an insertion statement for *Reg* is generated. Applying target egds may result in losing entities of source because inserting a tuple into a table may not be possible due to integrity constraints. For example, since information about students and departments is stored in the same table with *stName* as a primary key, existence of a department depends on existence of a student. In our example, *e3* cannot be inserted to *ST* because there is no student who is assigned to this department. This is a trade-off between data consistency and data completeness where a designer may relax some target egds to gain complete data exchange. However, the algorithms we propose priorities integrity constraints and does not allow breaking any target egd constraint.

If attacker sends query to the server, the server analyses the attacker by putting more queries to confirm whether he is an attacker or an honest participant. If he is attacker, server provides the noise added attribute to the attacker.

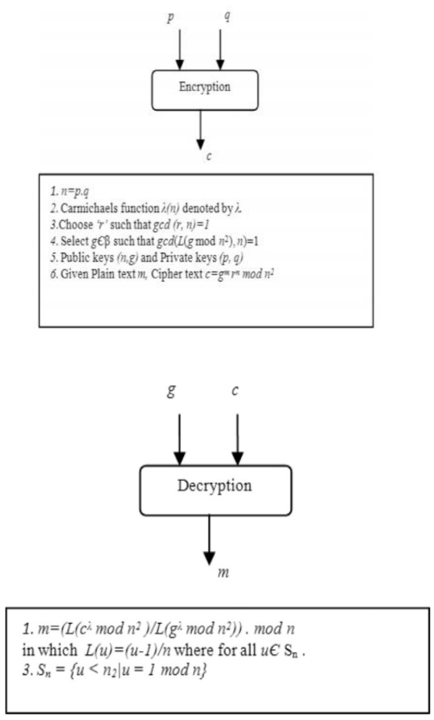
III. PROPOSED WORK

2.1 Paillier encryption

The Paillier is a probabilistic public-key algorithm with homomorphic property. The computation of the *n*th residue classes is considered to be a difficult process. Pascal Paillier has proposed a public key encryption scheme [23] on composite residuosity classes. The scheme is based on the problem of ensuring the number is an *n*th residue modulo *n*2. The encryption scheme comprises modular exponentiation, multiplication, and some specific operations and is shown in Fig. 1. First the spider web construction algorithm is developed and a sequence of numbers (S) is obtained. A plain-text *m* is encrypted, and the cipher text *c* is generated as given in Eq. (1), where $m, r < n$ *and* $m,r \varepsilon S$,

$$c = g^m r^n \bmod n^2 \qquad (1)$$

The value $g^m \bmod n^2$ is pre-computed, and the result is multiplied by $r^n \bmod n^2$. The same computation time is achieved using two modular exponentiation modules. However, pre-computation does not affect the security of the system, provided that the chosen value *g* fulfills the requirement $g \ \varepsilon \ \beta$ imposed by the setting of the paillier scheme. To further improve the security in the proposed scheme, we presented Spider Embedded Paillier (SEP) cryptography system. In this system, the random number g is not selected as wish. The spider web construction algorithm is developed and a sequence of numbers (S) is obtained. From sequence S, g is selected in such a manner that it must satisfy the setting prescribed in the paillier scheme. It should be in the range of 0 to n. Hence the random number g must be in the form that $g \ \varepsilon \ \beta; \ g \ \varepsilon \ (S)$. The random number r is selected from the sequence generated from spider with condition prescribed in paillier scheme. $r \ \varepsilon \ (S);$ $0<r<n$. the algorithm form spider web construction is illustrated subsequent sections.Decryption (*D*) of paillier homomorphic cryptography (PHC) requires a modular exponentiation of bases *c* and *g*. Computation is improved through precalculation of the modular inverse of *L* ($g\lambda$ mod *n*2) because *g* and λ are unique values for one key. A cipher text *c* is decrypted, and *m* is obtained. This is depicted in Fig. 2. The initial attempt on designing hardware architecture for a paillier based homomorphic encryption using field-programmable gate array (FPGA) [5] involves shift registers, an Arithmetic & Logic Unit (ALU) and a control unit. The exploitation of parallelism of the paillier includes deep pipeline ALU to achieve high clock frequency, decrease in data dependencies, resource sharability to achieve high throughput design along with the consideration of both latency and circuit size. In the proposed method of paillier encryption, the CPAS processing element is designed for achieving good quality metrics.



1. $n=p.q$
2. Carmichaels function $\lambda(n)$ denoted by λ.
3. Choose 'r' such that $gcd\ (r, n)=1$
4. Select $g \in \beta$ such that $gcd(L(g \bmod n^2), n)=1$
5. Public keys (n,g) and Private keys (p, q)
6. Given Plain text m, Cipher text $c=g^m r^n \bmod n^2$

1. $m=(L(c^{\lambda} \bmod n^2)/L(g^{\lambda} \bmod n^2)) . \bmod n$
in which $L(u)=(u-1)/n$ where for all $u \in S_n$.
3. $S_n = \{u < n_2 | u = 1 \bmod n\}$

Figure 1. 2: Identity-Based Encryption (IBE)

Identity-Based Encryption (IBE) is an alternative of public key encryption. The major advantage is simplified key management in a certificate based Public Key Infrastructure (PKI) by using human-unique identities for e.g., name, email address, IP address, etc. as public keys in Private Key Generator. In the IOT environment the devices are the users. After generation of keys the Private Key Generator communicates to the user (IOT device or the mobile device). Because of this, the physical devices are able to stay linked. If both devices are authenticate PKG serves the device2 keys. The message (M) can be transfer to device1 to device2 is shown in Fig. 2

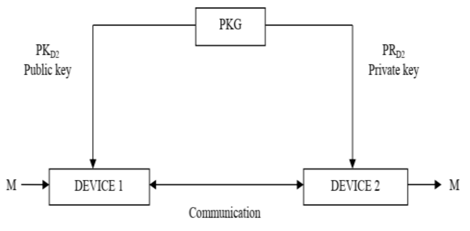


Fig. 2. IOT device-to-device

During a misdeed, i.e., when any device if found to misbehave, then the central key management system (the PKG) will revoke the keys of the misbehaving IOT device. This is known as revocation and is shown in Fig. 3. Normally, all the users despite the consequences of whether their keys have been revoked or not, have to make contact with PKG frequently to prove their identities and update new private keys. It requires that private key generator is online and the protected channel must be maintained for all communication.

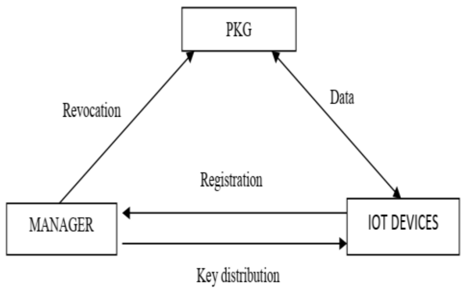


Fig. 3. Revocation process in IOT

Key Update Cloud Service Provider (KU-CSP) is a central system used for outsourced revocation. This is shown in Figure 4. The KUCSP updates the users key each and every time when ever the user needs. KU-CSP mostly operates key generation related operations during key-update and key-issuing processes. KU-CSP leaving only a invariable number of uncomplicated operations for PKG and users to do locally.

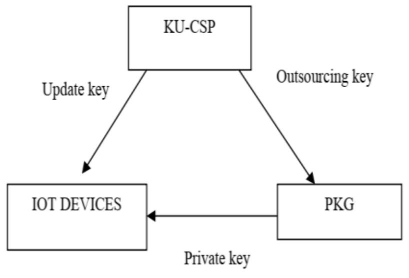


Fig. 4. System model of IBE with outsourced revocation for IOT environment

2.2 Spider embedded Paillier encryption

Generally the spider species construct web for catching its prey. The spider web is built by the spider silk, which is thrust out from its spinnerets. Every spider has some glands and each gland is able to produce spider silk. The diameter of silk in the spider web is in the range of 0.02mm-0.15 mm. The spider web becomes visible to human eye due to the reflection of light on the silk. A spider with a body length of 5mm can build a web for a volume of 100m^3. Spider web can be built in vertical or horizontal planes. There are many kinds of spider web. Of them orb web is built in the vertical plane and sheet web is built in the horizontal plane. Fig.1.2 shows sheet web, orb web and a hypothetical, vertical web built between two plants. In web construction, the spider first creates a sticky silk thread and thrust it out in the air to stick onto a suitable place. Then the spider moves along the thread to the fixed place and adds a thread to provide more strength to the thread. Then it again comes back to the original position and builds other threads to form a web. The different kinds of web like orb web, cobweb, sheet web can be constructed based on the type of spider and based on the approach of silk fixing.

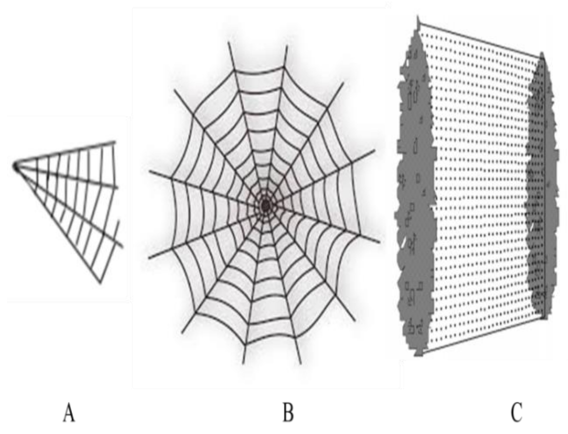


Figure III.1: Spider web A) Sheet web B) Orb web C) hypothetical vertical web

In constructing the orb web, the spider first emits a sticky silk thread from its mouth and makes the thread to fuse over an appropriate place. Then the spider travels over the thread to reach that place by attaching one more silk thread to enhance the strength of the silk thread. Then it goes back to the first position and forms other threads to construct a web. Meanwhile, it makes the thread looses and make it to hang in the middle. Then spider reaches that position and fixes that loosened thread to a suitable point to form a 'Y' shape as depicted in figure 5. Then it attaches the thread from the last point to the first two points to form a triangle shape as depicted in figure 5. Then from the center of the triangle, spider forms radial lines using the thread as in figure 6. After forming the radial lines, it again weaves the spiral lines by choosing the centre point as reference (as shown in figure 7). The number of radial lines and spiral lines can be varied by varying the angle between the two consecutive radial lines and the distance between the two consecutive spiral lines respectively. The point of intersection of the spiral lines and the radial lines with the spiral lines are taken as keys for the security pattern. The algorithm for constructing the orb web is explained in the next section.

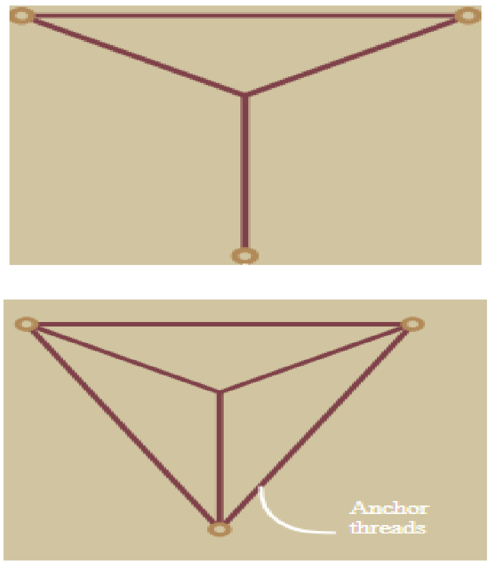


Figure III.2: Orb web frame formation

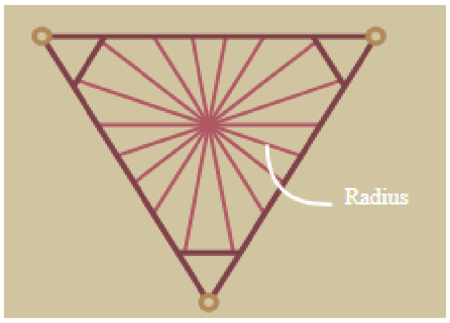


Figure III.3: forming radial lines in the frame

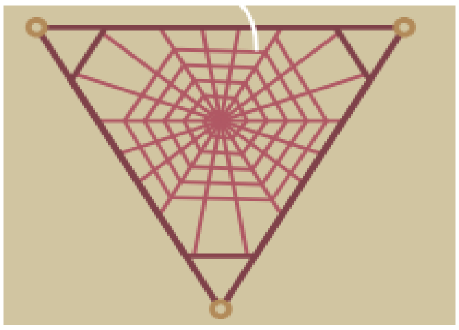


Figure III.4: Formation of spiral lines

2.2.1 Algorithm for spider security pattern generation based on orb web construction

Based on the orb web construction steps in spider, the algorithm is proposed to construct the artificial orb web. In this work, the uniform orb web is selected. The proposed algorithm consists of four major processes, which are the construction of initial radial lines, other radial lines and frame, initial spiral line and other spiral lines. In this method, the centre position and the number of radial lines can be changed as per the requirement. The length of radial lines determines the size of the frame. As in figure 5, three initial radial lines are formed to form 'Y' shape. Then other radial lines are constructed based on equal angular displacement. The spiral lines are then drawn over the radial lines. The distance between the spiral lines can be changed. This process is accomplished through the four steps given below.

2.2.1.1 Radial lines construction

Step-1: In this step, the initial radial lines are constructed using yshape () function. In this function, using three initial radial lines 'Y' shape is formed at first (see figure 8a). The angle between them is computed by dividing 360^o by 3. Hence the angle between two lines will be 120^o. The first radial line (OA) is drawn with angular displacement of 60^o from the y-axis in the first quadrant. The second radial line (OB) is drawn with angular displacement of 60^o from the y-axis in the second quadrant. The third radial line (OC) is placed on the negative side of the y-axis.

<u>Y-shape algorithm</u>

1. input: xmax, ymax, xmin=0, ymin=0, d_i, R, δ, line=1,α
2. yshape()// initial lines in 'Y' shape

```
3.  angle =30// for line OA
4.  draw (angle)
5.  angle =150// for line OB
6.  draw (angle)
7.  angle=270// for line OC
8.  draw (angle)
9.  end
```

Step-2: Then equal partitioning () function is applied using which a new radial line is drawn between the two existing radial lines. New radial line O1 is drawn on the positive side of the y-axis, O2 is drawn in the third quadrant with the displacement of angle 30^o from the x-axis. Similarly, O3 is drawn in the fourth quadrant with the displacement of angle 30^o from the x-axis. Now the radial lines are equally spaced with the displacement of 60^o. In this step three new radial lines are drawn (see figure 8b).

Equal partitioning algorithm

```
1.  equalpartitioning()//other radial lines
2.  angle=90// for line O1
3.  draw (angle)
4.  angle=210// for line O2
5.  draw (angle)
6.  angle=330// for line O3
7.  draw (angle)
8.  angle=60// for line O1'
9.  draw (angle)
10. angle=120// for line O2'
11. draw (angle)
12. angle=180// for line O3'
13. draw (angle)
14. angle=240// for line O4'
15. draw (angle)
16. angle=300// for line O5'
17. draw (angle)
18. angle=0// for line O6'
19. draw (angle)
20. end
```

Step-3: The space between the old radial line and the new radial line is partitioned using equal partitioning function again. Radial line O1' is drawn in the first quadrant in between OA and O1. Radial line O2' is drawn on the second quadrant in between OB and O1. Radial line O3' is drawn on the negative side of y-axis. Radial line O4' is drawn in the third quadrant in between OC and O2. Radial line O5' is drawn in the fourth quadrant in between OC and O3. Radial line O6' is drawn on positive x-axis (in between OA and O3). Now six new radial lines are drawn and there are totally 12 lines that are displaced by 30^o with each other. This process continues until when the convergence reaches. The minimum displacement angle should converge to the factor α which is given as input. Based on the convergence factor, the strength of the security pattern is determined. The displacement angle (θ) is calculated as below.

$$\boldsymbol{\theta} = \frac{\mathbf{360}}{\boldsymbol{n_i}}\ (\mathbf{3})$$

Where n_i denotes the number of radial lines in iteration i.

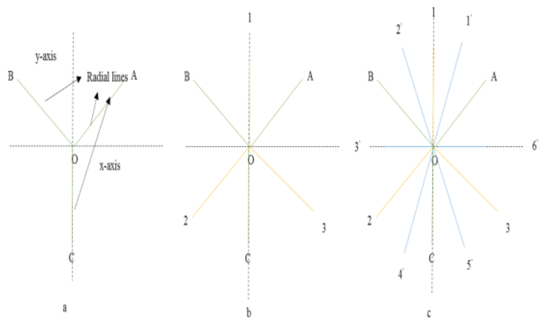


Figure III.5: Radial line construction a) Formation of 'Y' shape b) and c) formation of radial lines

draw () function

```
1.  draw(angle)// pass the angle as parameter in draw() function
2.  x=ymax/tan(angle)// x-point of line
3.  plot([xmin, ymin], [x,ymax])// draw line from O to the respective point
4.  line=line+1
5.  if (line%3)==0)
6.  partangle=360/line
7.  end
8.  if(partangle>=α)
9.  return
10. end
11. end
```

2.2.1.2 Spiral line construction

In constructing the spiral lines, the spiral () function is used in which the initial distance, d_i from the origin point 'O' and the fixed radius R are given as input. The spider moves from 'O' on the line OA by the distance d_i. Then the spider moves on the anti-clock wise direction and the distance is increased by δ. Therefore the spider intersects the radial line at distance d_i+δ from 'O'. This distance is increased by δ in each move until there d_{new} reaches R. The calculation of d_{new} is given below.

$$\boldsymbol{d_{new}} = \boldsymbol{d_i} + \boldsymbol{\delta}\ (\mathbf{4})$$

$$\boldsymbol{d_{new}} = \boldsymbol{d_{new}} + \boldsymbol{\delta}\ (\mathbf{5})$$

The spider lines are drawn as shown in figure 9.

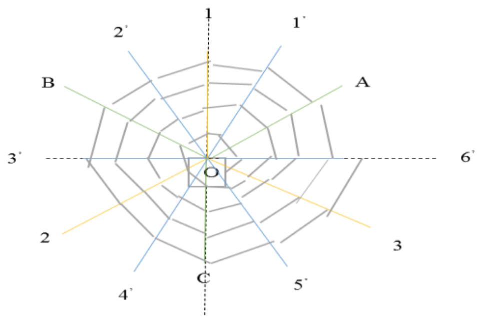


Figure III.6: Orb web construction

After constructing the orb web completely, the x-axis values are taken as time in ms and the y-axis values are taken as amplitude. The x and y values corresponding to the intersection points are taken and the security pattern is formed and it is embedded with Paillier encryption. The same parameters, which are used for orb web construction the transmitter side, is used at the receiver for building the web in the matching pattern, to facilitate the recovery of the original message. The algorithm for steps involved in the orb web construction is given below.

Spiral line function

```
1.  spiral (di, R, δ)
2.  x1=di,y1=0
3.  angle=30
4.  di=dnew
5.  dnew=dnew+δ
6.  if (dnew<=R)
7.  y2=dnew
8.  x2=dnew/tan(angle)
9.  plot([x1,y1], [x2,y2])// draw spiral line from one
    radius to another radius
10. x1=x2, y1=y2
11. go to step 5
12. end
```

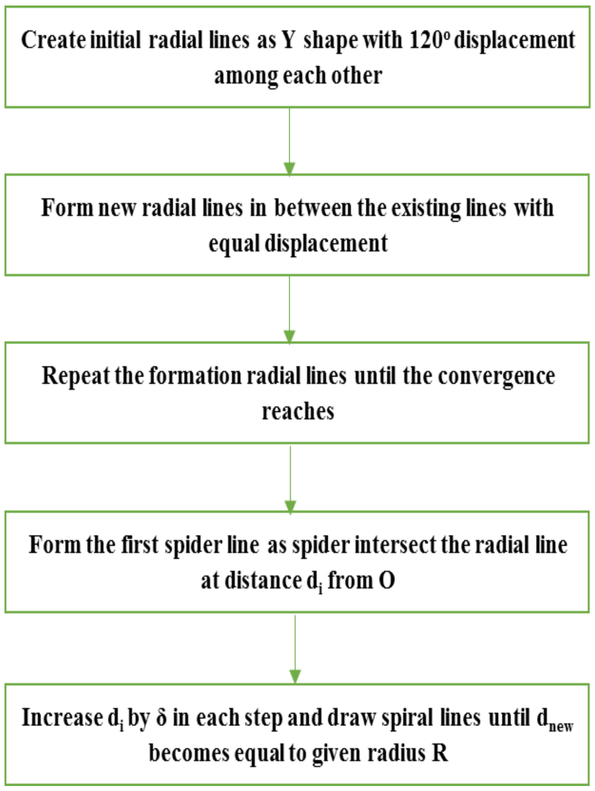

Create initial radial lines as Y shape with 120° displacement among each other
Form new radial lines in between the existing lines with equal displacement
Repeat the formation radial lines until the convergence reaches
Form the first spider line as spider intersect the radial line at distance d_i from O
Increase d_i by δ in each step and draw spiral lines until d_{new} becomes equal to given radius R

Figure III.7: Flow chart of the proposed spider algorithm

IV. PERFORMANCE EVALUATION

The proposed spider embedded Paillier encryption scheme is evaluated by comparing the parameters with the Homomorphic encryption based horizontal partitioning approach, conventional Paillier encryption and spider embedded Paillier encryption scheme. In order to evaluate the performance metrics the proposed method is implemented for heterogeneous databases. We considered three databases, namely, SQL server, MySQL server and MS Access.

Table 9 illustrates the mining time of the proposed system and its variation on the usage of heterogeneous DBs and homogeneous DBs. In addition the mining time is compared by employing different number of records; 500, 1000 and 15000. When 500 records are employed, 3 DBs are used. When 1000 records are employed, 4 DBs are used. When 1500 records are employed, 5 DBs are used. Figure 5.6 shows the comparison mining time required for different number of DBs and different kind of DBs. When number of DBs used is more, the time consumed for data mining is more and viceversa. Similarly the time consumption is less in Heterogeneous database comparing with the Homogeneous database. The mining time decreases by a maximum of 20.8% in heterogenous DBs than in homogeneous DBs. The mining time consumed in the application of Homogeneous DBs is 270 ms, 612 ms and 1814 ms when the number of DBs involved are 3,4 and 5 respectively. The mining time consumed in the application of Heterogeneous DBs is 230 ms, 612 ms and 1605 ms when the number of DBs involved are 3, 4 and 5 respectively.

Table 1: Mining Computation of Homogeneous and Heterogeneous Databases

Total Records	Homogeneous				Heterogeneous			
	Total DB	Mining Time (Sec.)			Total DBs	Mining Time (Sec.)		
		homomorphic	Paillier	Spider+ Paillier		homomorphic	Paillier	Spider+ Paillier
500	3	290 ms	293	270	3	240 ms	242	230
1000	4	630 ms	628	612	4	567 ms	565	554
1500	5	1830 ms	1825	1814	5	1630 ms	1633	1605

Table 10 describes the mining time required for different DBs. Also it is validated by changing the number of attributes in DB. When the numbers of attributes are less, the mining time is less. Thus the time taken for data mining depends on the number of attributes involved in the DBs and also the number of DBs. Figure 5.7 illustrates the same. The mining time is 103 ms, 348 ms and 493 ms when we employ 2 DBs with different number of attributes of 10, 20 and 30. The mining time is 332 ms, 447 ms and 830 ms when we employ 3 DBs with different number of attributes of 10, 20 and 30. The mining time is 572 ms, 639 ms and 1030 ms when we employ 5 DBs with different number of attributes of 10, 20 and 30. The mining time increases with more DBs as well as it increases when the number of attributes increases.

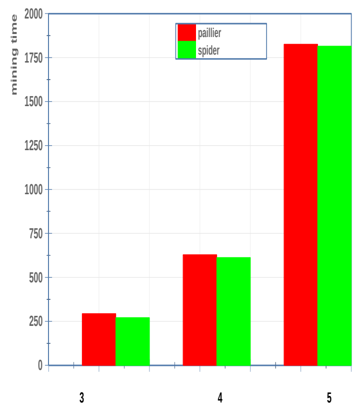


Figure IV.1: Comparison of mining time for homogeneous DBs and for different number of DBs

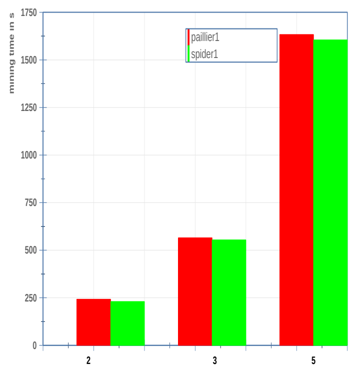


Figure IV.2: Comparison of mining time for homogeneous DBs and for different number of DBs

Table 2: Attributed based Computation to Achieve Privacy

Number of Attributes	Number of DBs		
	2	3	5
10	90 ms	322 ms	562 ms
20	337 ms	437 ms	629 ms
30	485 ms	823 ms	1009 ms

Table 11 shows the analysis of time consumption for various scenarios. The mining time consumption is compared by changing the size of the key and by changing the number of encrypted items. The mining time is examined by changing the key size and for various number of encrypted items. Figure 5.8 depicts variation of mining time with respect to key size for different numbers of items encrypted. Mining time is less than 0.0003s for key size of 256 and with 1K of encrypted items. When we employ 10K of encrypted items, the mining time is increased to 10s. When the encrypted items size is increased by 10 times to 100K, the time of 90s is consumed for mining. When the encrypted items size is increased by 10 times to 1000K, the mining time also increases by 10 times and it is 900s. When the key size is increased the mining time also increases correspondingly. The mining time consumption for 512 bits of key size is 5s, 47s, 487s and 4760s for the size of encrypted items of 1K, 10K, 100K and 1M respectively. Analogously, the mining time consumption for 1024 bits of key size is 29s, 286s, 2930s and 28962s for the size of encrypted items of 1K, 10K, 100K and 1M respectively. Thus it is observed that when the size of encrypted items increased by 10 times, the time also increased by 10 times approximately. The transfer time is analysed by changing the number of encrypted items

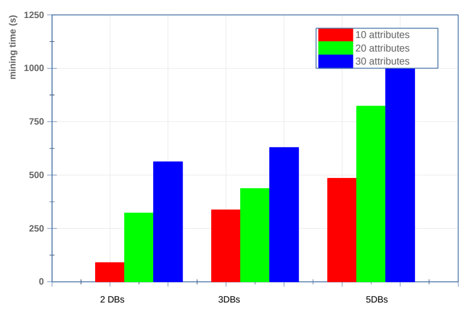


Figure IV.3: comparison of mining time for different DBs with different number of attributes

Table 3: Comparison of mining time for different Number of Items Encrypted and for different key size

Number of Items Encrypted	Key Size (Sec.)			Transfer Time (Sec.)
	256	512	1024	
1k	0.0001	4	25	0.0034
10k	9	45	278	0.0082
100k	85	477	2850	0.077
1M	875	4670	28562	0.35

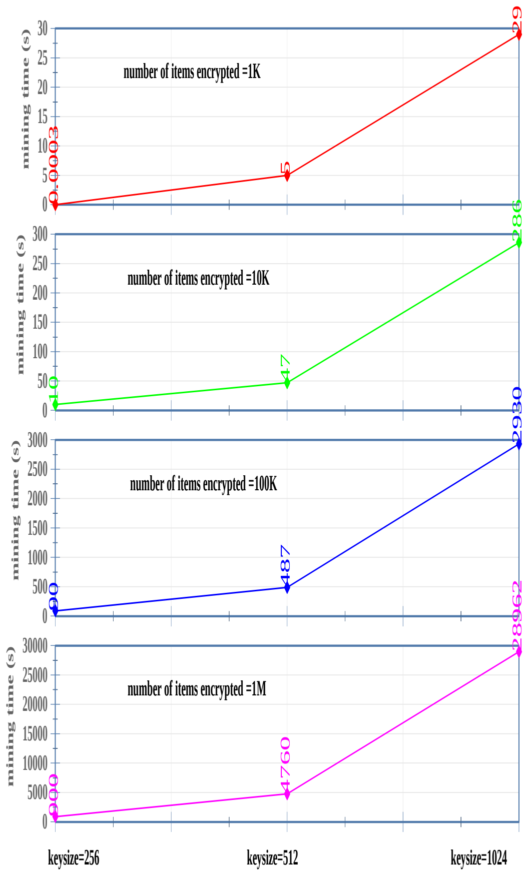


Figure IV.4: Variation of mining time with respect to key size for different numbers of items encrypted

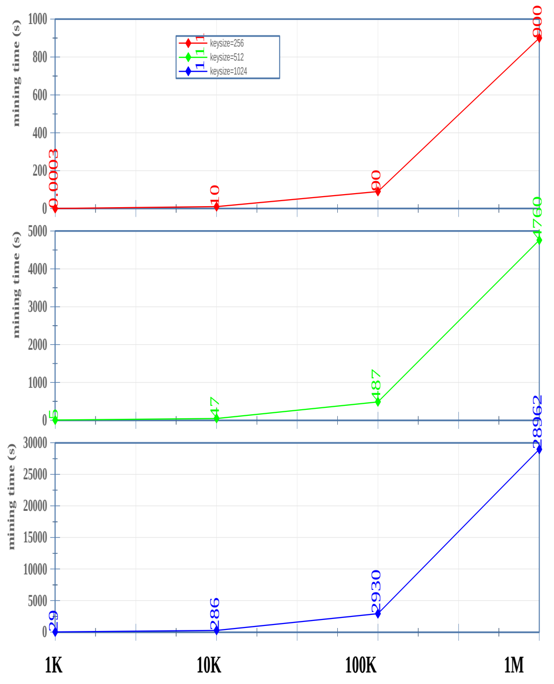


Figure IV.5: Variation of mining time with respect to numbers of items encrypted for different key size

Figure 5.11 depicts variation of mining time with respect to numbers of items encrypted for different key size. Transfer time is 0.0038s when 1K of encrypted items are involved. It is 0.009s, 0.08s and 0.41s when the numbers of encrypted items involved are 1k, 10k, 100k and 1M respectively.

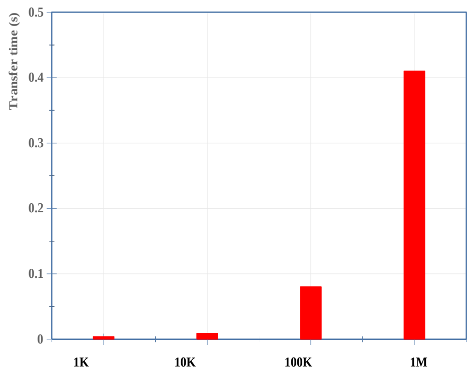


Figure IV.6: Comparison of transfer time for different numbers of items encrypted

V. CONCLUSION

The spider web construction algorithm is developed and a sequence of number is obtained. In constructing the orb web, the spider first emits a sticky silk thread from its mouth and makes the thread to fuse over an appropriate place. Then the spider travels over the thread to reach that place by attaching on more silk thread to enhance the strength of the silk thread. Then it goes back to the first position and forms other threads to construct a web. Then the spider embedded paillier encryption scheme is presented to enhance security and the proposed system is evaluated for homogenous and heterogeneous DBs. Based on the web construction step in spider, the algorithm is proposed to construct the artificial orb web. In this work, the uniform orb web is selected. The proposed algorithm consists of four major processes, which are the construction of initial radial line, other radial line and frame, initial spiral line and other spiral line.

REFERENCES

1. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. *Proc. 20th Int. Conf. on very Large Databases (VLDB 1994, Santiago de Chile)*, 487–499. Morgan Kaufmann, San Mateo, CA, USA 1994
2. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo. Fast Discovery of Association Rules. In [10], 307–328.
3. M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'97, Newport Beach, CA)*, 283–296. AAAI Press, Menlo Park, CA, USA 1997
4. J. Han, H. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. *Proc. Conf. on the Management of Data (SIGMOD'00, Dallas, TX)*, 1–12. ACM Press, New York, NY, USA 2000
5. J. Vaidya and C. Clifton, Privacy preserving association rule mining in vertically partitioned data, in Proc. SIGKDD, 2002, pp. 639–644
6. Sheikhalishahi, Mina, and Fabio Martinelli. "Privacy preserving clustering over horizontal and vertical partitioned data." *Computers and Communications (ISCC), 2017 IEEE Symposium on*. IEEE, 2017.
7. Gudes, Ehud, and Boris Rozenberg. "Collaborative Privacy Preserving Frequent Item Set Mining in Vertically Partitioned Databases." *Data and Applications Security XVII*. Springer, Boston, MA, 2004. 91-104.
8. Kantarcioglu, Murat, and Chris Clifton. "Privacy-preserving distributed mining of association rules on horizontally partitioned data." *IEEE transactions on knowledge and data engineering* 16.9 (2004): 1026-1037.
9. Yin, Xiaoxin, and Jiawei Han. "CPAR: Classification based on predictive association rules." *Proceedings of the 2003 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2003.
10. Zaki, Mohammed Javeed, et al. "New Algorithms for Fast Discovery of Association Rules." *KDD*. Vol. 97. 1997.
11. Han, Jiawei, et al. "Mining frequent patterns without candidate generation: A frequent-pattern tree approach." *Data mining and knowledge discovery* 8.1 (2004): 53-87.
12. Agrawal, Rakesh, and Ramakrishnan Srikant. "Fast algorithms for mining association rules." *Proc. 20th int. conf. very large data bases, VLDB*. Vol. 1215. 1994.
13. Brossette, Stephen E., et al. "Association rules and data mining in hospital infection control and public health surveillance." *Journal of the American medical informatics association* 5.4 (1998): 373-381.
14. Li, Lichun, et al. "Privacy-preserving-outsourced association rule mining on vertically partitioned databases." *IEEE Transactions on Information Forensics and Security* 11.8 (2016): 1847-1861.
15. "Executive order – promoting private sector cybersecurity information sharing." https://www.whitehouse.gov/the-press-office/2015/02/13/executive-order-promoting-private-sector-cybersecurity-information-shari
16. M. E. Locasto, J. J. Parekh, A. D. Keromytis, and S. J. Stolfo, "Towards collaborative security and p2p intrusion detection," in Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC, pp. 333–339, IEEE, 2005.
17. Popa, L., Velegrakis, Y., Hernandez, M. A., Miller, R. J., and Fagin, R.: Translating Web data. Proc. Very Large Data Bases, 598-609(2002)
18. Miller, R. J., Haas, L. M., and Hernadez, M. A.: Schema mappings query discovery. Proc. Very Large Data Bases, 77-88 (2000)
19. X, Liang, Z. Cao, R. Lu, and L. Qin, "Efficient and secure protocol in fair document exchange," Comput.Stand. Interfaces, pp. 167–176. 2008.
20. J. A. Garay, R. Gelles, D. S. Johnson, A. Kiayias, and M. Yung, "A little honesty goes a long way –The two-tier model for secure multiparty computation," TCC 2015, Part I, LNCS 9014, pp. 134-158, 2015.]

21. Harshita and S. Tanwar. "Implementation of fair exchange and anonymous online e-cash protocol for e-commerce," I.J. Information Technology and Computer Science, pp. 66-74 , August 2016.
22. S. Ketchpel, "Transaction protection for information buyers and sellers," Paper presented in the Proceedings of the Dartmouth Institute for Advanced Graduate Studies 95, Electronic Publishing and the Information Superhighway, Boston, USA., 1995.
23. Alaraj and M. Munro, "An e-commerce fair exchange protocol that enforces the customer to be honest," Int. J. Product Lifecycle Manag. pp. 114–131, March 2008
24. A . Alaraj and M. Munro, "An efficient e-commerce fair exchange protocol that encourages customerand merchant to be honest," In Computer Safety, Reliability, and Security; Lecture Notes in Computer Science; Springer Berlin Heidelberg, Berlin, Germany, pp. 193–206.2008.
25. S. Coretti, J. Garay, M. Hirt, and V. Zikas, "Constant round asynchronous multiparty computation based on one way functions," ASIACRYPT 2016, Part II, LNCS 10032, pp. 998–1021, 2016.
26. Vakilinia, D. K. Tosh, and S. Sengupta, "Attribute based sharing in cybersecurity information exchange framework," in *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2017 International Symposium on*, IEEE, 2017.
27. P. Porras and V. Shmatikov, "Large-scale collection and sanitization of network security data: risks and challenges," in *Proceedings of the 2006 workshop on New security paradigms*, pp. 57–64, ACM, 2006.
28. E. Adar, "User 4xxxxx9: Anonymizing query logs," in Proc of Query Log Analysis Workshop, Intl. Conference on World Wide Web, 2007
29. C. V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," *Computers & Security*, vol. 29, no. 1, pp. 124–140, 2010.
30. M. E. Locasto, J. J. Parekh, S. Stolfo, and V. Misra, "Collaborative distributed intrusion detection," 2004.
31. S. Suthaharan, "Big data classification: Problems and challenges in network intrusion prediction with machine learning," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 4, pp. 70–73, 2014.