jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2020, VOL. 66, NO. 4, PP. 637-645
Manuscript received April 1, 2020; revised October, 2020. DOI: 10.24425/ijet.2020.134022

Integer Matrix Keys for Secure Data Aggregation in Clustered Wireless Sensor Networks

G. Chethana, and K.V. Padmaja

***Abstract*—Providing Privacy and security for aggregated data in wireless sensor networks has drawn the attention of practicing engineers and researchers globally. Several cryptographic methods have been already proposed to solve security and data integrity problems for aggregated data. Matrix cryptography is a better option for creating secure encryption/decryption algorithms to counter quantum attack. However, these algorithms have higher computational cost and increased communication overhead. Hence, a new technique of loss-less secure data aggregation in Clustered Wireless Sensor Networks is presented. The proposed method uses integer matrices as keys for data security and data integrity. Matrix operations are carried out in finite field Z_p. Loss-less secure data aggregation is extended for homomorphic summation while the cipher text expansion ratio is kept substantially low. The proposed algorithm has inbuilt fast and efficient signature verification facility. The execution time of our signature verification mechanism is found to be approximately 50 percent less compared to a couple of standard existing signature verification schemes.**

***Keywords*—loss-less data aggregation, integer matrices as keys, finite field Zp, homomorphic aggregation**

I. Introduction

THE main task of Data Aggregation (DA) in WSN is to combine the collected data from the sensor nodes into a meaningful aggregate [1-2]. The aggregate may be sum, average, count, min, max, median or any other aggregating function of the individual data. The aggregate type depends on the requirement of the users. In general, during the aggregation process, DA eliminates trivial, duplicate and redundant data values. DA effectively reduces the data size from the aggregator to the final End User (EU) through intermediate Relay Nodes (NSs), Base Station (BS), and Cloud Server (CS). The reduction in the data size in turn decreases the traffic load and consequently the computational time and energy consumption are also reduced. This increases the life of the Wireless Sensor Network (WSN).

A. Secure Data Aggregation

With DA, the aggregated data is located at the aggregator and at the intermediate relay nodes. If these locations are attacked, the entire data will be compromised. The exposure of aggregated data is more detrimental than the leakage of individual data. Thus, the data security is very critical when DA is implemented compared to the non-aggregated data transmission. Secure Data Aggregation (SDA) provides both aggregation and security for the aggregate. The main security features are as follows. Data confidentiality (privacy): In WSN, data confidentiality ensures the secrecy of the sensed data. It should not be disclosed to unauthorized agents. Data confidentiality is an important ingredient in defense related (military) and medical applications. Data confidentiality is provided using suitable cryptographic schemes. Data integrity: Data integrity provides protection against data alteration due to noise, communication channel errors or active attacks, etc. Normally, Message Authentication Codes (MAC) and error detection codes are employed to provide data integrity. Data authenticity: Data authenticity ensures the data source is genuine and not fake. Compromised source nodes with fake identities are detected with appropriate digital signature scheme.

B. Lossy and Loss less Secure Data Aggregation

SDA can be lossy or loss-less. In lossy SDA, some of the information, which is insignificant for the concerned application, is lost. But the contextually essential data is retained. On the other hand, in loss-less SDA, the full data is retained. The final decoder can recover the full data from the encrypted and aggregated data.

II. Related work

Several review articles [2-15] are available on SDA in WSN. In these papers, the authors have comprehensively described various methods for SDA for different topological configurations. Different types of security requirements and various cryptographic techniques to meet the above requirements suitable for SDA are discussed in these survey papers. Homomorphic encryption/decryption schemes for SDA are discussed by a few authors [16-19]. In [16], the authors Domingo-Ferrer [19] have used privacy homo-morphism method for SDA. Here the sensor data is split into several components at encryption and then recovered by combining them after decryption. This process increases the overall computational cost. In [17], Niu et al., have described lossy data aggregation with integrity scheme using bilinear maps. The disadvantage of this scheme is loss of data during aggregation and heavy computational overhead due to the use of bilinear maps. In [18], homomorphic Paillier encryption for protecting data privacy and homomorphic MAC to provide data integrity are adopted. This scheme involves the calculation of modular exponentiations and modular inverses which result in excessive computational overhead when the data from large number of nodes are to be aggregated. In [20-21], the authors use the principle of CDMA to aggregate the

G. Chethana, Research Scholar, is with the dept. of Electronics and Communication Engg, at RV College of Engg,, VTU University, Belagavi, Karnataka, India, (e-mail: chethanag@rvce.edu.in).
K.V. Padmaja is with the dept. of Electronics and Instrumentation Engg, RV College of Engg B'lore VTU University, Belagavi, Karnataka, India, (e-mail: padmajakv@rvce.edu.in).

© The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0, https://creativecommons.org/licenses/by/4.0/), which permits use, distribution, and reproduction in any medium, provided that the Article is properly cited.

data. Here Message Authentication Codes are used for verifying data integrity. This increases the computational cost when the size of the data set is large. In [22-23], lattice (matrix) based cryptographic methods are used for secure data aggregation. Here low valued noise and scrambling matrices are used to randomize the encryption. Selection of these matrices is critical in the design of encryption/decryption process. The lengths of the public keys are relatively large. In [22-23] signature insertion/verification is substantially complex. In [30] and [31], matrix methods are used to provide digital signature. But when adopted for secure data aggregation, the length of the signature increases linearly with the number of data elements to be aggregated. To overcome the above disadvantages, we propose a new method of SDA that uses integer matrices as keys with easy signature insertion and verification facility.

III. PROPOSED WORK

Our proposed method provides Loss-Less SDA using Matrices as Keys. The method is designated as SDA-MK. Here the individual data from the sensors are combined and encrypted to get the secure aggregated data which is decrypted by the final receiver. SDA-MK uses integer matrices in the finite field (Galois field) Z_p. In Z_p, all the elements are in the range 0 to $p-1$ and all the algebraic/arithmetic operations are carried out with respect to modulo p which is relatively a large prime number.

A. Symbols, Notations and definitions

The basic layout of the WSN is shown in Fig.1. It has a single Cluster Head (CH) which collects the data from the sensors and then aggregates the data and forwards the aggregated data to the Cloud Server (CS) via intermediate Relay Nodes (RNs) and Base Station (BS). The End User (EU) gets the aggregate from CS. Here the CH acts as the loss-less Aggregator. EU acts as de-aggregator. The number of sensors nodes attached to the CH is taken as *N*.

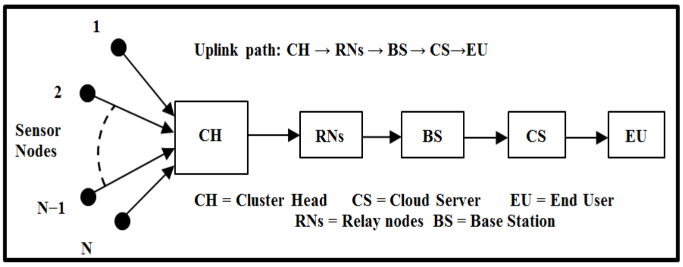
1
2
Sensor Nodes
N−1
N
Uplink path: CH → RNs → BS → CS→EU
CH
RNs
BS
CS
EU
CH = Cluster Head CS = Cloud Server EU = End User
RNs = Relay nodes BS = Base Station

Fig.1 Basic Layout of SDA

B. Sensor Data Row Vector

Let the Individual sensor data received by the CH be denoted by d_1, d_2… d_N in appropriate units. Then the sensor data is represented by the Sensor Data Row Vector ***D*** as,

$$\mathbf{D} = [d_1, d_2, \ldots, d_N] \qquad (1)$$

The size of ***D*** is 1x*N*. Individual sensor data values d_1, d_2… d_N are assumed to be integers in the range 0 to $p-1$. Here p is the modulus of the finite field Z_p. {If the original data from a node is a fractional number, it is converted into the corresponding integer by scaling up by 10, 100, or say 1000.Thus 2.34 → 2.34*100 = 234. At the receiving side it is scaled down by 100}. The modulus p is a prime number greater than the estimated maximum element of the data vector ***D.*** The elements of ***D*** belong to Z_p.

*C. Secrete Key Matrix **C***

Matrix ***C*** is the secret key of the SDA system. The elements of ***C*** belong to the finite field Z_p. The size of **C** is *M* x (*N*+1) with *M* > *N*+1.Thus $C \in Z_p^{M\times(N+1)}$. The rank of **C** should be *N*+1. The reason for choosing a rectangular matrix for **C** of rank *N*+1, instead of a square matrix, is explained in section III E.

D. Base Matrix B

Base Matrix ***B*** is the Modular Matrix Inverse [24] of ***C***. The size of **B** is (*N*+1) x *M* and $\boldsymbol{B} \in Z_p^{(N+1)\times M}$. Thus,

$$\boldsymbol{B} = \boldsymbol{mmi(C}, p) \qquad (2)$$

Here, function **mmi**() stands for the modular matrix inversion with respect to the modulus *p*. From the definition of **mmi**(),

$$\boldsymbol{mmi(C}, p)\boldsymbol{*C} = \boldsymbol{I}_{(N+1)\mathrm{x}(N+1)} \qquad (3)$$

Since ***C*** is a tall matrix (No. of rows > No. of columns) of rank *N+1*, it has the left inverse [25] as,

$$\boldsymbol{C}_{\boldsymbol{left}}^{-1} = (\boldsymbol{C^T} * \boldsymbol{C})^{-1} * \boldsymbol{C^T} \qquad (4)$$

Here, $\boldsymbol{C^T}$ is the transpose of ***C***. The size of the product $(\boldsymbol{C^T} * \boldsymbol{C})$ is {(*N*+1) xM} x {*M* x (*N*+1)} = (*N*+1) x (*N*+1) its rank is *N*+1. Therefore the inverse $(\boldsymbol{C^T} * \boldsymbol{C})^{-1}$exists. All the algebraic operations of (4) are carried out in Z_p. Then, **mmi**(**C**, p) can be expressed as,

$$\boldsymbol{mmi}(\boldsymbol{C}, p) = \boldsymbol{C}_{\boldsymbol{left}}^{-1} = (\boldsymbol{C^T} * \boldsymbol{C})^{-1} * \boldsymbol{C^T} \ \ (in\ mod\ p) \qquad (5)$$

The condition for the existence of the modular matrix inverse of $(\boldsymbol{C^T} * \boldsymbol{C})$ is $gcd(det(\boldsymbol{C^T} * \boldsymbol{C}), p) = 1.$ since, *p* is chosen to be a prime, and this condition is automatically satisfied. From (2) and (3),

$$\boldsymbol{B*C} = \boldsymbol{I}_{(N+1)\mathrm{\ x\ }(N+1)} \text{ (mod } p) \qquad (6)$$

Equation (6) can be rewritten as,

$$mod\ (\boldsymbol{B*C}, p) = \boldsymbol{I}_{(N+1)\mathrm{\ x\ }(N+1)}$$

When there is no ambiguity, with matrix multiplication implemented in Z_p, the above Equation can be simply written as,

$$\boldsymbol{B*C} = \boldsymbol{I}_{(N+1)\mathrm{\ x\ }(N+1)} \qquad (7)$$

In (7), the rectangular matrix ***B*** is called the Moore–Penrose inverse of ***C.*** The Moore–Penrose inverse of matrix ***C*** is unique [26] for a given ***C*** and p. In our proposed scheme, 'generalized inverse' of **C** is used instead of Moore–Penrose inverse. The generalized inverse is not unique and this property is utilized to randomize our encryption to prevent *Chosen Plaintext Attack* as will be described in section VII.B.

E. Generalized Inverse of C

The size of $\boldsymbol{C^T}$ *is* (*N*+1) x*M* with (*N*+1) < *M*. It is a fat matrix. Therefore it has modular null space [27]. Let the modular null space of matrix $\boldsymbol{C^T}$ be denoted by the matrix $\boldsymbol{F}$. Then, by the definition of modular null space, Matrix $\boldsymbol{F}$ in Z_p satisfies,

$$\boldsymbol{C^T*F} = \boldsymbol{0}_{(N+1)\text{ x }L} \tag{8}$$

$\boldsymbol{F}$ is obtained by using the standard function ModNull(C^T, p). The size of $\boldsymbol{F}$ is MxL where,

$$L = M - (N+1) \tag{9}$$

The RHS of (8) is an all zero matrix of size (N+1)xL. Taking the transpose of (8), we have,

$$\boldsymbol{F^T*C} = \boldsymbol{0}_{L\text{ x }(N+1)} \tag{10}$$

Now, consider the matrix **R** derived from the base matrix $\boldsymbol{B}$ and $\boldsymbol{F^T}$ as,

$$\boldsymbol{R} = \boldsymbol{B}+\boldsymbol{Y*F^T} \tag{11}$$

Where, $\boldsymbol{Y}$ is an arbitrary matrix of size (N+1)xL *in* Z_p. The size of $\boldsymbol{F^T}$ *is* LxM. The size of the product $\boldsymbol{Y*F^T}$ as well as $\boldsymbol{R}$ is (N+1) xM. In (11) addition and multiplication are carried out in Z_p. Now the product $\boldsymbol{R*C}$, from (11) will be,

$$\boldsymbol{R*C} = (\boldsymbol{B}+\boldsymbol{Y*F^T})\boldsymbol{*C} = \boldsymbol{B*C} + \boldsymbol{Y*F^T*C} \tag{12}$$

From (6), $\boldsymbol{B*C} = \boldsymbol{I}$ and from (10), $\boldsymbol{F^T*C} = 0$. Therefore, from (12),

$$\boldsymbol{R*C} = \boldsymbol{I}_{(N+1)\text{ x }(N+1)} \tag{13}$$

Equation (13) can be expressed in terms of the rows and columns of $\boldsymbol{R}$ and $\boldsymbol{C}$, respectively as,

$$\begin{bmatrix} \boldsymbol{R_1} \\ \boldsymbol{R_2} \\ . \\ . \\ \boldsymbol{R_N} \\ \boldsymbol{R_{N+1}} \end{bmatrix} * [\boldsymbol{C_1} \quad \boldsymbol{C_2} \quad . \quad . \quad \boldsymbol{C_N} \quad \boldsymbol{C_{N+1}}] =$$

$$\begin{bmatrix} \mathbf{1} & \mathbf{0} & . & . & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & . & . & \mathbf{0} & \mathbf{0} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \mathbf{0} & \mathbf{0} & . & . & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & . & . & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

This can be expressed as,

$$\boldsymbol{R_i} * \boldsymbol{C_j} = \begin{cases} 0 & if\ i \neq j \\ 1 & if\ i = j \end{cases} \tag{14}$$

Since, matrix $\boldsymbol{R}$ satisfies (13), it is the generalized inverse of $\boldsymbol{C}$**.** From (11), we see that matrix $\boldsymbol{R}$ depends on the arbitrary matrix $\boldsymbol{Y}$**.** Therefore, $\boldsymbol{R}$ is not unique and it can take a large number of multiple values depending on the selected dissimilar values of $\boldsymbol{Y}$. In this paper, these multiple distinct random $\boldsymbol{R}$'s are designated as $\boldsymbol{R}\{1\}$, $\boldsymbol{R}\{2\}$… $\boldsymbol{R}\{i\}$ and so on. Realization of multi-valued $R\{i\}$'s is possible due to the existence of the modular null space matrix $\boldsymbol{F}$ whose size is MxL. Therefore, for the existence of $\boldsymbol{F}$, the value of L should be greater than 0. That is, from (9), M > (L+1) which means the number of rows of $\boldsymbol{C}$ should be greater than the number of columns of $\boldsymbol{C}$. If $\boldsymbol{C}$ is a square matrix, this condition is not satisfied and we cannot have multi-valued $\boldsymbol{R}\{i\}$'s.Let $\boldsymbol{R\{i\}}$ be the i^{th} instance of $\boldsymbol{R}$, Then, from (13),

$$\boldsymbol{R\{i\}*C} = \boldsymbol{I}_{(N+1)\text{x}(N+1)} \tag{15}$$

For i = 1, 2… so on. In general, when there is no ambiguity, we use the symbol $\boldsymbol{R}$ to represent any one version of $R\{i\}$'s. The multi-valued property of $\boldsymbol{R}$ is an essential requirement to prevent chosen plaintext attack as will be explained in section VII.B

F. Key setup and Distribution

Key Setup and Distribution is implemented by the Key Generation Center (KGC). A suitable, relatively large prime number p is selected as the modulus for Z_p. Here, p is the security parameter. Higher the value of p, greater is the security. Since, modular arithmetic is used throughout all the operations, all operand values should be in the range 0 to $p-1$. Hence the selected p should be greater than the estimated maximum value, say d_{max} of the data set. Then $d_{max} < p$ and all d's belongs to Z_p. Hence, the data sequence $\boldsymbol{D}$ gets aggregated correctly.

G. Selection of Matrix C and Digital signature Parameter S

After selecting p, the secret key matrix $\boldsymbol{C}$ of size (N+1)xM is generated randomly by KGC, such that the elements of $\boldsymbol{C}$ are in $\boldsymbol{Z_p}$. Parameter N is same as the number of sensors assigned to the CH**.** The extra row of $\boldsymbol{C}$ is used for signature verification as will be explained later. The value of M is chosen to be greater than (N+1). Here, M is taken as $M = N+1+L$ where L is in the range 2 to 4. A higher value of L increases the size of the keys and cipher text which in turn increases the communication overhead. A smaller value of L decreases the security level. While selecting matrix $\boldsymbol{C}$, it should be ensured that its modular inverse $\boldsymbol{B}$ exists.

The digital signature parameter, represented by S is a scalar in $\boldsymbol{Z_p}$**.** In general, it is relatively a large number in $\boldsymbol{Z_p}$ so that it is difficult to predict by a hacker.

H. Calculation of B, F and R

After selecting $\boldsymbol{C}$, its modular matrix inverse $\boldsymbol{B}$ is calculated using the function $\boldsymbol{B} = \boldsymbol{mmi(C,\ p)}$ as given by (2). Standard built functions based on **Gauss Jordan row reduction method** and extended Euclid's algorithm are available [24], [28] to find $\boldsymbol{B}$. The modular null space $\boldsymbol{F}$ *of* $\boldsymbol{C}$ is also calculated based on the row reduction method [29]. From $\boldsymbol{B}$ and $\boldsymbol{F}$**,** $\boldsymbol{R\{i\}}$'s are calculated as given by (11).

I. Key Distribution

The calculated values of $\boldsymbol{R\{i\}}$*'s* along with the signature parameter S and the modulus value p are sent by the KGC to the CH through a secured channel. The CH encrypts and aggregates using these values. The KGC also sends $\boldsymbol{C}$**,** S and p to the End User for decryption/de-aggregation.

IV. PROPOSED SYSTEM MODEL

The basic model of the Secure Data Aggregator and De-aggregator is shown in Fig.2. In the layout shown in Fig. 2, the intermediate relay nodes (RNs), BS and the Cloud Server (CS) do not have access to $\boldsymbol{R}$ $\boldsymbol{\{i\}}$'s**,** signature parameter S and p.

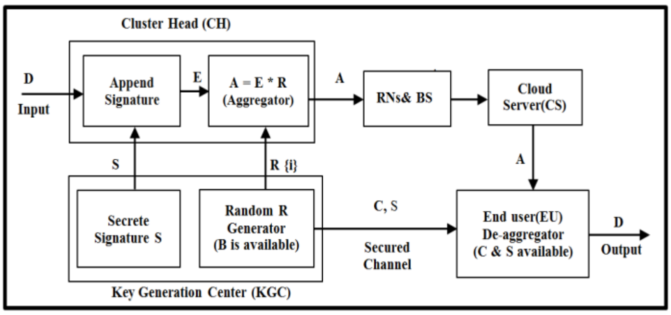

Fig.2 Basic model of the Secure Data Aggregator and De-aggregator

But the End User (EU), which is the de-aggregator, has access to these parameters.

A. Secured Loss less Data Aggregation with digital signature at CH

On receiving all data values from the sensors (over one TDMA cycle), the CH formulates the data vector as,

$\boldsymbol{D} = [d_1, d_2 \ldots d_N]$

The size of $\boldsymbol{D}$ is 1xN. Then, CH appends the signature parameter S to $\boldsymbol{D}$ to get the augmented data vector $\boldsymbol{E}$ as,

$$\boldsymbol{E} = [\boldsymbol{D}, S] = [d_1, d_2 \ldots d_N, S] \tag{16}$$

Now, the size of $\boldsymbol{E}$ is 1x (N+1). The secure aggregate vector, designated by $\boldsymbol{A}$ is generated as,

$\boldsymbol{A} = mod\ (\boldsymbol{E*R\{i\}}, p)$ can be rewritten as,

$$\boldsymbol{A = E*R\{i\}} \tag{17}$$

Size of $\boldsymbol{A}$ *is* (1x (N+1)) x ((1+N) xM) = 1xM. Here, matrix **R{*i*}** is an instance of multivalued matrix ***R***. Row vector ***A*** is sent to CS through RNs and the BS. The EU receives aggregate A from the CS. Here, row vector ***A*** is the encrypted aggregate of data vector ***E.*** Therefore, from the encryption point of view, ***A*** is the cipher text and ***E*** is the plain text.

B. Signature Verification and De-aggregation at EU

On receiving the encrypted aggregate ***A***, the EU verifies the signature and if successful, de-aggregates ***A***.

1) Signature Verification

In SDA-MK, the signature parameter S is embedded in aggregate vector ***A*** and S is recovered from ***A*** as follows.

At EU, consider the product W = ***A*C***. From (17), substituting for ***A***, we get,

$$\boldsymbol{W=A*C=E*R\ \{i\}*C} \tag{18}$$

Here, size of ***A*** is 1xM and size of ***C is*** M x (N+1). Therefore size of ***W is*** 1x (N+1). From (13), (18) and (16),

$$\boldsymbol{W = E*R\{i\}*C = E*I = E} = [d_1, d_2 \ldots d_N, S] \tag{19}$$

From (18) and (19),

$$\boldsymbol{A*C} = [d_1, d_2 \ldots d_N, S] \tag{20}$$

Expressing ***C*** in terms of its columns, (20) can be expressed as,

$$\boldsymbol{A*[C_1, C_2 \ldots C_N,}\ C_{(N+1)}\boldsymbol{]} = [d_1, d_2 \ldots d_N, S] \tag{21}$$

From (21), $\boldsymbol{A}*C_{(N+1)} = S$. Therefore the calculated value of S recovered from ***A*** by EU is,

$$S_{cal} = \boldsymbol{A}*C_{(N+1)} \tag{22}$$

The EU calculates S_{cal} using (22) and verifies the signature by comparing S_{cal} with S which has been already received from KGC. If $S_{cal} \neq S$, the signature verification fails. When the signature verification fails, the authenticity/integrity of the received aggregate ***A*** is lost. Therefore, received ***A*** is discarded without de-aggregation.If $S_{cal} = S$, the signature verification is successful. The signature verification at EU provides source and data authenticity. Then the EU de- aggregates the data as follows.

2) De-aggregation/Decryption at EU

From (21), taking the first N elements from RHS and LHS, we get,

$$[d_1, d_2 \ldots d_N] \boldsymbol{= A*[C_1, C_2 \ldots C_N] = D} \tag{23}$$

Since, matrix ***C*** is available to EU, the EU de-aggregates/decrypts ***A*** to recover original data ***D*** using (23).

C. Authentication and data integrity with signature

The aggregator CH attaches the signature S which is known to CH and EU only. When there is no error or alteration in ***A,*** signature S_{cal} should be same as the original S. On the other hand, during aggregation and transmission, if ***A*** gets corrupted or altered (say by a malicious attacker or noise etc.), S_{cal} would be different from S. Thus, our signature verification scheme is unforgeable. The example given below demonstrates Signature verification, De-aggregation and authentication process in WSN.

*Example*1: p = 499; N = 4; M = 7; L=M−N−1=2. *All operations are in* $\boldsymbol{Z_P}$. *Matrix* **C** *is generated randomly. From* **C,** *matrix* ***R****'s are obtained for different values of random matrix* **Y**, *two such samples of R's are given below.*

Matrix **C** = $\begin{bmatrix} 56 & 21 & 68 & 70 & 39 \\ 71 & 06 & 60 & 45 & 94 \\ 30 & 45 & 03 & 16 & 98 \\ 52 & 03 & 56 & 55 & 68 \\ 90 & 46 & 26 & 79 & 91 \\ 90 & 65 & 42 & 31 & 85 \\ 13 & 28 & 29 & 23 & 38 \end{bmatrix}$

Matrix **R{1}** = $\begin{bmatrix} 91 & 62 & 471 & 286 & 221 & 136 & 56 \\ 91 & 17 & 332 & 454 & 112 & 220 & 469 \\ 312 & 45 & 287 & 381 & 263 & 220 & 58 \\ 246 & 233 & 360 & 137 & 283 & 358 & 104 \\ 168 & 349 & 154 & 215 & 199 & 475 & 202 \end{bmatrix}$

Matrix **R{2}** = $\begin{bmatrix} 45 & 496 & 414 & 375 & 378 & 472 & 004 \\ 399 & 350 & 015 & 451 & 004 & 59 & 390 \\ 221 & 455 & 351 & 001 & 295 & 189 & 459 \\ 259 & 452 & 323 & 149 & 113 & 228 & 45 \\ 109 & 150 & 362 & 70 & 406 & 360 & 200 \end{bmatrix}$

It can be verified that ***R***{1}****C* = *I*** *and* ***R***{2}****C* = *I***. In this example, let ***R*** = ***R***{1}**.** Data vector ***D*** is taken as, ***D*** = [7 23 74 76]**.** Signature S is taken as, S = 27. Augmented data is E = [7 23 74 76 27] = **[*D*, *S*]**

Encrypted Aggregate ***A*** = ***E*R*** is found to be, ***A*** = [148 348 316 468 67 449 386].

Signature verification: The last column of ***C*** is C_{N+1} = [39 94 98 68 91 85 38] ^{T}. Then, S_{cal} = ***A****C_{N+1}, found to be same as, *S* = 27.Then, the data vector ***D*** is recovered as ***D* = *A****[C_1, C_2… C_N] = [7 23 74 76].

V. SECURED SUM AGGREGATION

In this section security for the sum is achieved using SDA-MK approach. The sum aggregate provides the overall quantitative status of the sensed data values. Secured Sum Aggregation (SSA) generates the aggregate that represents the sum of data values. From the sum of data values average values can be easily determined. Let the time slots at which sensor readings are taken be denoted by t_1, t_2,…,t_K where $t_1 < t_2 < \ldots < t_K$. Let us assume that the time slots are uniform. Total number of time slots taken is K. For brevity, time slot corresponding to t_j be referred as time slot *TS*(*j*). Let the data value generated by sensor *i* at *TS*(*j*) be denoted by *d*(*i*, *j*). The data values of *N* sensors form data matrix D at time slots *TS*(*j*) for *j* = 1 *to K*, is augmented with signature parameter *S* for each *j*, to get the augmented matrix ***Q*** of size (*N*+1)x*K* and the data matrix D is represented in yellow background as shown in Table I. *U*(*j*)*'s & V*(*i*)'s are columns and rows of matrix ***Q*** respectively as in Table I.

TABLE I
AUGMENTED MATRIX ***Q*** OF SIZE (*N*+1) x*K*

TS(j)s Sensors		*TS*(1)	*TS*(2)	…	*TS*(*j*)	…	*TS*(*k*)
		U(1)	*U*(2)	…	*U*(*j*)	…	*U*(*K*)
Sensor 1	*V*(1)	*d*(1,1)	*d*(1,2)	…	*d*(1 *j*)	…	*d*(1, *K*)
Sensor 2	*V*(2)	*d*(2,1)	*d*(2,2)	…	*d*(2,*j*)	…	*d*(2, *K*)
….	…	…	…	…	…		…
Sensor i	*V*(i)	*d*(*i*, 1)	*d*(*i*, 2)	…	*d*(*i*, *j*)	…	*d*(*i*, *K*)
…	…	…	…	…	…	…	…
Sensor N	*V*(*N*)	*d*(*N*,1)	*d*(*N*,2)	…	*d*(*N j*)	…	*d*(*N*, *K*)
Signature	*V*(*N*+1)	*S*	*S*	…	*S*	…	*S*

Here, column *U*(*j*), corresponding to time slot *TS*(*j*) is given by,

$\boldsymbol{U}(j) = [d(1, j), d(2, j), \ldots, d(i, j), \ldots, d(N, j), S\,]^T$ (24)

for *j* = 1 *to K*. Column vector *U*(*j*) gives the data values of all the *N* sensors at time slot *TS*(*j*) appended with *S*. Then matrix *Q* can be expressed as,

$$\mathbf{Q} = [\boldsymbol{U}(1),\ \boldsymbol{U}(2),\ \ldots,\ \boldsymbol{U}(j),\ \ldots,\ \boldsymbol{U}(k)] \quad (25)$$

A. Temporal Data Summation

Consider the Sum of Columns of ***Q*** represented by ***SU***. From (24),

$$\boldsymbol{SU} = \boldsymbol{U}(1) + \boldsymbol{U}(2) + \ldots + \boldsymbol{U}(j) + \ldots + \boldsymbol{U}(K) \quad (26)$$

Here, ***SU*** gives the sum of Columns of matrix **Q**. That is,

$$\boldsymbol{SU} = \begin{bmatrix} d(1,1) \\ d(2,1) \\ \cdot \\ d(i,1) \\ \cdot \\ d(N,1) \\ S \end{bmatrix} + \begin{bmatrix} d(1,2) \\ d(2,2) \\ \cdot \\ d(i,2) \\ \cdot \\ d(N,2) \\ S \end{bmatrix} + \cdots + \begin{bmatrix} d(1,j) \\ d(2,j) \\ \cdot \\ d(i,j) \\ \cdot \\ d(N,j) \\ S \end{bmatrix} + \cdots$$

$$+ \begin{bmatrix} d(1,K) \\ d(2,K) \\ \cdot \\ d(i,K) \\ \cdot \\ d(N,K) \\ S \end{bmatrix}$$

Column vector ***SU*** can be expressed as,

$$\boldsymbol{SU} = \begin{bmatrix} \sum_{j=1}^{K} d(1,j) \\ \sum_{j=1}^{K} d(2,j) \\ \cdot \\ \sum_{j=1}^{K} d(i,j) \\ \cdot \\ \sum_{j=1}^{K} d(N,j) \\ K * S \end{bmatrix} \quad (27)$$

The term $\sum_{j=1}^{K} d(i,j)$ represents the sum of the elements of row $\boldsymbol{V}(i)$ of matrix Q. Summation $\sum_{j=1}^{K} d(i,j)$ is the temporal sum of data of sensor *i*. Let us represent the temporal sum by *sv*(*i*) as,

$$sv(i) = \sum_{j=1}^{K} d(i,j) \quad (28)$$

Then, from (27) and (28),

$$\boldsymbol{SU} = [sv(1), sv(2) \ldots sv(i)., sv(N), sv(N+1\,)]^T \quad (29)$$

Here, *sv* (*N*+1) is the sum of the (*N*+1)th row of *Q*, which is equal to *K***S*. From (29) and (26),

$$\boldsymbol{SU} = [sv(1),\ sv(2) \ldots, sv(i) \ldots, sv(N),\ sv(N+1)]^T =$$

$$U(1) + U(2) + \cdots + U(j) + \cdots + U(K) \quad (30)$$

SU is a column vector of size (*N*+1) x 1. The ith element of SU, represented by *sv* (*i*) gives the temporal sum of the data values of sensor *i*, *for i* = 1 *to N*. The value of *sv* (*N*+1) = *K* **S*, where *S* is signature parameter as in Table I.

B. Secured Sum Aggregation to get temporal sum

Consider the summation represented by **ATU** i.e, Aggregate of Total of U's as,

$$\boldsymbol{ATU} = U(1)^T * R\{1\} + U(2)^T * R\{2\} + \ldots + U(j)^T * R\{j\} + \ldots. + U(K)^T * R\{K\} \quad (31)$$

Here, the size of *U* (*j*) ^{T} *is* 1x (*N*+1) and that of *R* {*j*} is (*N*+1)x*M*. The size of **ATU** is 1x*M*. In (31), matrix ***R*** {*j*} is the *j*th generalized inverse of **C**. Therefore, for *j* = 1 *to K*,

$$R\{j\} * C = \boldsymbol{I}_{(N+1)\,\mathrm{x}\,(N+1)} \quad (32)$$

Temporal sum block diagram is shown in Fig. 3.The CH executes the summation given by (31) to encrypt as well as to aggregate *U (i)*'s. At the CH, the sensor data vectors U (1) arrives at *TS* (1), *U*(2) arrives *at TS*(2),…, *U(K)* arrives at *TS(K)*. Therefore, the time interval between adjacent time slots is the delay between the arrivals of the consecutive *U (j)*'s. Hence the CH can do summation cumulatively one term at a time as,

```
ATU = 0;
for i = 1 to K
ATU = ATU+U(i)$^T$ *R{i}; //all operations
                              are in $Z_p$
End of for
```

C. Decryption at End User

Decryption of ***ATU*** is carried out at EU to get the decrypted output ***DTU*** (De-aggregate of Total of ***U***'s) as,

$$\boldsymbol{DTU} = \boldsymbol{ATU} * \boldsymbol{C} \tag{33}$$

Correctness of decryption: From (31) and (33),

$$\boldsymbol{DTU} = (\boldsymbol{U}(1)^T * \boldsymbol{R}\{1\} + \boldsymbol{U}(2)^T * \boldsymbol{R}\{2\} + .. + \boldsymbol{U}(j)^T * \boldsymbol{R}\{j\} + \ldots + \boldsymbol{U}(K)^T * \boldsymbol{R}\{K\}) * \boldsymbol{C}$$
$$= \boldsymbol{U}(1)^T * \boldsymbol{R}\{1\} * \boldsymbol{C} + \boldsymbol{U}(2)^T * \boldsymbol{R}\{2\} * \boldsymbol{C} + \ldots + \boldsymbol{U}(j)^T * R\{j\} * \boldsymbol{C} + \ldots + \boldsymbol{U}(K)^T * \boldsymbol{R}\{K\} * \boldsymbol{C}$$

From (32), $\boldsymbol{R}\{j\} * \boldsymbol{C} = \boldsymbol{I}$ for all j's, we have,

$$\boldsymbol{DTU} = \boldsymbol{U}(1)^T + \boldsymbol{U}(2)^T + \ldots + \boldsymbol{U}(j)^T + \ldots + \boldsymbol{U}(K)^T = [\boldsymbol{U}(1) + \boldsymbol{U}(2) + \ldots + \boldsymbol{U}(j) + \ldots + \boldsymbol{U}(K)]^T \tag{34}$$

The size of row vector ***DTU*** *is* 1x *(N*+1*)*. From (34) and (30),

$$(\boldsymbol{DTU})^T = \boldsymbol{U}(1) + \boldsymbol{U}(2) + \ldots + \boldsymbol{U}(j) + \ldots + \boldsymbol{U}(K) = \boldsymbol{SU}$$

$$= [sv(1),\ sv(2) \ldots, sv(i) \ldots, sv(N),\ sv(N+1)]^T \tag{35}$$

From (35),

$$\boldsymbol{DTU} = [sv(1),\ sv(2) \ldots, sv(i) \ldots, sv(N),\ sv(N+1)] \tag{36}$$

The i^th^ element, ***DTU*** (*i*) gives *sv*(*i*) *for i* = 1 *to N*. That is the row vector ***DTU***(1*: N*) gives the temporal sum of *N* sensors. In other words,

$$\boldsymbol{DTU}(1{:}\ N) = Temporal\_sum\_N \tag{37}$$

The last element ***DTU*** (*N*+1) gives the sum of *K* number of signature parameter S (refer Table I) which is equal to *K*S*.

That is, $\boldsymbol{DTU}\ (N+1) = sv(N+1) = K*S \quad (38)$

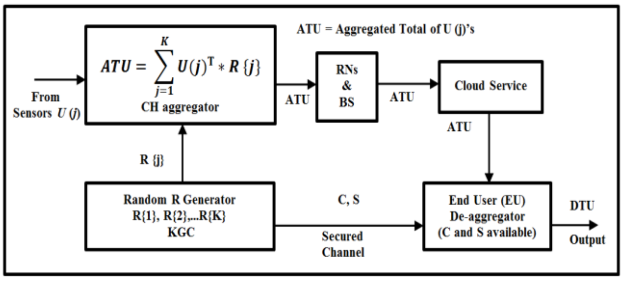

Fig.3 Secure data aggregation to get the temporal sum

D.Signature verification

Considering equation (33), ***DTU*** =***ATU*C***, Where the size of ***DTU*** is 1x (*N*+1).

Secrete key Matrix **C** can be expressed in terms of its sub matrices as

$$\boldsymbol{C} = [\boldsymbol{G}, H] \tag{39}$$

Where ***G*** = [C1, C_2…., C_N] i.e., first *N* columns of ***C*,** and $H = C_{N+1}$ i.e., last column of ***C***.

From (33) and (39),

$$\boldsymbol{DTU} = \boldsymbol{ATU} * [\boldsymbol{G}, H] = [\boldsymbol{ATU} * \boldsymbol{G}, \boldsymbol{ATU} * H] \tag{40}$$

Here, the size of *G is M*x*N* and that of *H is M*x1. In (40), the size of *ATU*G is* (1x*M)*x*(M*x*N)* = 1x*N*. The size of ***ATU*****H* is (1x*M) x (M*x1) = 1x1. Therefore, the first N elements of row vector **DTU** is given by,

$$\boldsymbol{DTU}\ (1{:}\ N) = \boldsymbol{ATU} * G \tag{41}$$

From (41) and (37),

$$Temporal\_sum\_N = \boldsymbol{ATU} * G \tag{42}$$

The last element **DTU** is,

$$\boldsymbol{DTU}\ (N+1) = \boldsymbol{ATU} * H \tag{43}$$

From (43) and (38), we see that

$$\boldsymbol{ATU} * H = K * S \tag{44}$$

Here, signature verification is done by checking that ***DTU*** (*N*+1) = *K*S.*

E. Verification and Decryption algorithm at End user

The inputs and output of the algorithms are:
Input: ATU, Secret key Components G and H as derived from C. [as in Eqn. (39)].
Output: Signature Verification and *Temporal_sum_N*

Algorithm 1:

```
 Receive ATU;
 Get ATU*H
 If ATU*H ≠ K*S //Signaure is incorrect
 Diplay "Signature Verification Failed"
 Discard ATU
 Request for re-transmission
 Exit
 Else
 Display "Signature Verification Success"
 Get sum of temporal data for N sensors
 as,
 Temporal_Sum_N = ATU*G     //sum ready
 Exit
endif
```

F. Constraints on Secured sum Aggregation

In all calculations of SSA aggregation and decryption, we use modular arithmetic. Therefore the result of decryption, **DTU** is in Z_p. From (36),

$$\boldsymbol{DTU}(i) = sv(i)\ mod\ p \quad , \text{for } i = 1 \text{ to } N+1 \tag{45}$$

If the exact sum *sv*(*i*) as given by (28) is less than *p*, then from (45) we see that $DTU(i) = sv(i)$. On the other hand if $sv(i) \geq p$, then *DTU*(*i*) does not give the exact sum *sv*(*i*) because of *mod* operation. Hence, for the satisfactory application of SSA, *sv* (*i*)'s should be less than *p* for *i* = 1 to *N*+1. That is

$$sv\,(i) < p, \text{ for } i = 1 \text{ to } N+1 \qquad (46)$$

By choosing a large *p*, equation (46) will be satisfied. The Aggregation and de-aggregation of temporal sums is demonstrated in example 2.

*Example 2****:*** *The data from 4 sensors, N = 4 for time slots, K = 3, along with Signature parameter S = 27 is shown in Table II. Matrix C, value of p, matrices R {1} and R {2} are same as in Example 1. Since the number of columns K of D is 3, we need R {3} which satisfies equation (15).Data matrix D is shown with yellow background.*

TABLE II
AUGMENTED DATA MATRIX Q

TS(j)s / sensors		TS(1)	TS(2)	TS(3)	Sum
		U(1)	U(2)	U(3)	SU
Sensor 1	V(1)	7	70	34	111
Sensor 2	V(2)	23	62	85	170
Sensor 3	V(3)	74	90	4	168
Sensor 4	V(4)	76	76	60	212
Signature	S	27	27	27	81

The matrix *R*{3} is found to be,

$$\text{Matrix } R\{3\} = \begin{bmatrix} 476 & 206 & 24 & 440 & 441 & 11 & 171 \\ 275 & 335 & 152 & 469 & 224 & 47 & 483 \\ 407 & 264 & 245 & 15 & 149 & 366 & 483 \\ 426 & 263 & 151 & 25 & 113 & 287 & 113 \\ 138 & 112 & 407 & 413 & 95 & 377 & 228 \end{bmatrix}$$

Aggregated Row Total ***ATU*** is obtained using (31) as,

ATU = [308 342 165 291 84 166 486]

From equation (44), ***ATU****H* is found to be, ***ATU****H* = 81 which is same as *K**S* = 3*27. Therefore, the signature is correct.
Temporal_sum_N from equation (42) is found as ***ATU****G* = [111 170 168 212] which is found equal to the temporal sum of 4 sensors as shown in Table II.

De-aggregation with tampered data: Let the attacker change **ATU** (1) to *309* as,
ATU*_tamp* = [309 342 165 291 84 166 486]. Then,
ATU*_tamp*H* is found equal to 120 which is unequal to the expected *K**S* which is 81. Here, the signature verification has failed. This inequality points out the loss of data integrity.

G. Spatial Data Summation

Consider the data matrix ***D***, shaded yellow in Table I. Each column of ***D*** gives the data values of all the *N* sensors at that time slot. Therefore the sum of the elements each column (column sum) of ***D*** gives the spatial sum at the corresponding time slot. Since the column sums of matrix ***D*** are same as the row sums of $\boldsymbol{D}^T$, the methods of section V.*A* can be used to find the spatial sums at successive time slots using matrix $\boldsymbol{D}^T$ instead of ***D.***

VI. AGGREGATION OF SUM BY HOMOMORPHIC ENCRYPTION

The aggregate summation described in section V can be interpreted in terms of homomorphic addition as follows. Consider the encryption operation described by (31)

$$\boldsymbol{ATU}=\boldsymbol{U}(1)^T*\boldsymbol{R}\{1\}+\boldsymbol{U}(2)^T*\boldsymbol{R}\{2\}+\ldots+\boldsymbol{U}(j)^T*\boldsymbol{R}\{j\}+\ldots.+\boldsymbol{U}(K)^T*\boldsymbol{R}\{K\}$$

Let us introduce the symbol ***ATU*** *(j)* to represent the product term ***U****(j)* ^{T} ****R***{j} for *j* = 1 *to K* as,

$$\boldsymbol{ATU}(j) = \boldsymbol{U}(j)^T*\boldsymbol{R}\{j\} \qquad (47)$$

The size of ***U***(j)^T is 1x (*N*+1) and that of ***R***{*i*} is (*N*+1)x*M*. The size of ***ATU****(j)* is 1x*M*. Here, ***ATU(j)*** is the encryption of the augmented data vector ***U****(j)* ^{T} in time slot *TS*(*i*) {refer Table I}.

A. *Homomorphic addition at cloud Server*

In the Homomorphic addition scheme, the CH encrypts ***U(j)*** ^{T} according to (47) and transmits ***ATU****(j) for j = 1 to K* the Cloud Server (CS). The basic layout is shown in Fig. 4. The CH encrypts each ***U****(j)*^{T} to get ***ATU****(j)* as given by (47) and forwards it to the CS.

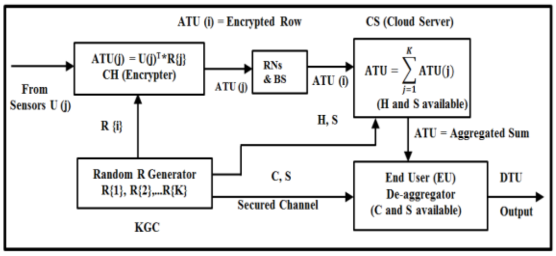

Fig.4. Homomorphic Addition in CS

B. *Signature Verification at CS*

The CS on receiving vector ***ATU*** *(j)*'s, checks for signature verification by calculating S_{cal} as,

$$S_{cal} = \boldsymbol{ATU}\,(j)*H \qquad (48)$$

where H is the last column of C i.e H = C $_{N+1.}$ In (48), the size of vector ***ATU*** (*j*) is 1xM and that of H is *M*x1. Equation. (48) is similar to (44). Verification is similar to as described in section V.*D*. Therefore, the decoding vector **H** and the signature parameter S are made available to CS at the start of the session. On successful verification, the CS adds the individual ***ATU*** (*j*)'s to get the sum **ATU** as,

$$ATU = \sum_{j=1}^{K} ATU(j) \qquad (49)$$

The EU on accessing ***ATU*** decrypts it to get ***DTU*** as described in signature verification and decryption algorithm under section V. Here, the CS acts as the homomorphic adder.

VII. SECURITY ANALYSIS OF SDA-MK

A. *Cipher text Expansion Ratio*

Cipher text Expansion Ratio (CER) expresses how many times bigger the size of the cipher text is compared to that of its plaintext. CER is the ratio of the cipher text size to its plaintext size. A higher value of CER means the computational cost and communication overheads are higher. Smaller the CER better is the encryption efficiency. CER is defined as,

$$CER = \frac{Size\ of\ Ciphertext}{Size\ of\ Plaintext} \tag{50}$$

In our SDA-MK approach, from (17), the size of the cipher text is 1xM and from (1), the size of the plaintext is 1xN. Therefore,

$$CER = \frac{1xM}{1xN} = \frac{M}{N} \tag{51}$$

In SDA-MK, in the light of (9), the cipher text length $M = N+L+1$. Therefore, the cipher text length varies linearly with N.

B. *Protection against chosen Plaintext Attack*

In Chosen Plaintext Attack (CPA), the attacker can chose his plaintexts arbitrarily and get the corresponding cipher texts and based on these observations, the attacker tries to access the secret key(s) of the encrypter. In SDA-MK, the encryption is randomized as it uses different random secret keys, $\boldsymbol{R}\{i\}$'s for successive encryptions. Therefore, SDA-MK is immune to CPA.

C. *Protection against chosen Cipher text Attack*

Chosen Cipher text Attack (CCA) tries to break the security of the decrypter (in this case, the de-aggregator). The attacker can chose different arbitrary cipher texts and can access the corresponding decrypted plaintexts. Using these values, the attacker tries to crack the decryption key(s). In SDA-MK, the digital signature scheme provides protection against CCA. Here, each cipher text input to the decrypter is evaluated (50) for valid signature S. If the signature check fails, there is no plaintext output. Hence the CCA attacker is unable crack the decrypter as most of the plaintext responses are empty.

VIII. COMPARISON OF SDA-MK WITH OTHER METHODS

A. *Signature Verification Efficiency of SDA-MK*

In SDA-MK, a single signature (scalar S) provides verification for N data values collected from the sensors (16).Therefore SDA-MK is highly efficient compared to those schemes where data vector of length N requires a signature vector of length substantially greater than 1. Theoretical efficiency of SDA_MK is compared with Lyubashevsky Micciancio Signature (LMS) method [30] and Gupta and Biswas (GUPTA) [31]. In LMS method, the size of the generator matrix is SGx(SG+1). The maximum data value can be ($p-1$). Therefore the number of bits required to represent a data value is *ceil* (*Log*$_2$ ($p-1$)) which is approximately *ceil* (*Log*$_2$ (p)). Theparameters taken for comparison are the computational costs of signature verification as well as the lengths of messages and signatures. The comparison values are shown in Table III. Here, N is the number of sensors, $M = N+L+1$ (9). (For large values of N, the value of M is approximately equal to N). The cost of one scalar multiplication in *Zp is* [*Log*$_2$ (p)]^2 bit operations.

TABLE III
EFFICIENCY COMPARISON OF SDA-MK ALGORITHM

Method	Message length(bits)	Signature Length in bits(s_n)	Signature Verification Computational Cost	
			Total no.of bit operations (t_{nb})	Normalized signature length=t_{nb}/s_n
SDA-MK	N* Log$_2$(p)	Log$_2$(p)	M*[Log$_2$(p)]^2	M*Log$_2$(p)
LMS	N* Log$_2$(p)	SG*Log$_2$(p)	2*SG*N* [Log$_2$(p)]^2	2*N* Log$_2$(p)
GUPTA	N* Log$_2$(p)	*2*N**Log$_2$(p)	N*N*[Log$_2$(p)]^2 + N* Log$_2$(p)	*½ [N** Log$_2$(p) + 1]

SG = Number of rows of the generator matrix in LMS.

From Table III, it can be seen that SDA-MK is theoretically more efficient in signature verification compared to LMS method but almost equal to GUPTA method.

B. *Experimental Results*

Simulation is carried out in Matlab to determine the execution times for signature verification in SDA-MK, LMS,GUPTA methods.For LMS method the generator matrix size parameter SG is selected as 10. Finite Field modulus is set at p = 499. The number of data elements is varied from N =100 to 500 in steps of 50. The Execution times of the three methods are shown in Table IV. Here, T_1, T_2 and T_3 are execution times in micro-seconds.

TABLE IV
EXECUTION TIME COMPARISON OF SDA-MK ALGORITHM

Algorithm Sensors	SDA-MK	LMS	GUPTA	% time Saving w.r.t LMS	% time Saving w.r.t GUPTA
N	T_1	T_2	T_3	S_2	S_3
100	17.1	35.2	66.6	51.4	74.3
150	19.5	36.5	60.7	46.6	67.9
200	21.6	40.9	62.4	47.2	65.4
250	25.1	44.4	61.7	43.5	59.3
300	28.1	46.9	64.8	40.1	56.6
350	30.2	51.2	70.2	41.0	57.0
400	35.1	55.7	73.4	37.0	52.2
450	37.6	58.7	82.6	35.9	54.5
500	39.3	61.6	78.1	36.2	49.7

Percentage savings in execution times w.r.t LMS and GUPTA method are respectively given by.

$S_2 = 100*(T_2–T_1)/T_2$, $S_3 = 100*(T_3–T_1)/T_3$

S_2 and S_3 values are shown in the last two columns of Table IV. The timing values of Table IV are plotted in Fig. 5.

From the Fig.5, it can be seen that SDA-MK takes lower execution time for signature verification. (The execution times are machine dependent and the result shown in Fig. 5 is useful for the purpose of comparison only). The zig-zag natures of the plots is due to the modular algebraic operations of the corresponding algorithms.

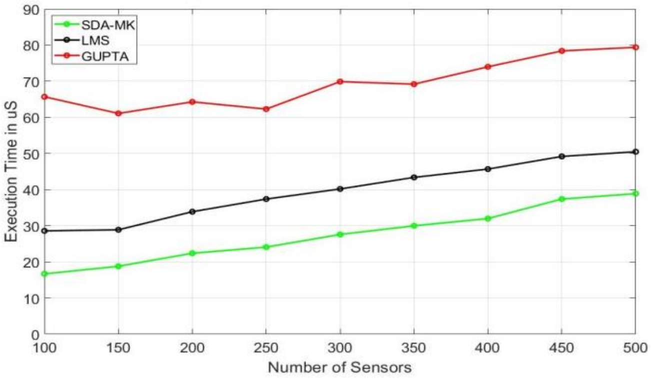

Fig.5.Execution Time versus Number of Sensor Nodes

IX. CONCLUSION

A new method of Loss-less Secure Data Aggregation using Matrix Keys is described. A speciality of this method is, the generation of multiple distinct random *orthogonal matrices to prevent chosen plain text attack***.** The novelty of this method is to achieve data privacy and data integrity simultaneously by embedding the digital signature in the aggregated data. The implementation of a new technique on sum aggregation is another major contribution of this research work. The encryption/decryption process has low computational/communication overhead. The digital signature also provides source authentication with unforgeability. The encryption process is inherently homomorphic for addition and SDA-MK is extended to aggregate the sum of sensor data values both temporally and spatially.

REFERENCES

[1] B. Krishnamachari, D. Estrin, and S. Wicker.”The impact of data aggregation in wireless sensor networks”. In Proc. Intl. Workshop of Distributed Event Based Systems, Proceedings of IEEE INFOCOM, New York, NY. July 2002.

[2] Randhawa, S.; Jain, S.” Data Aggregation in Wireless Sensor Networks: Previous Research, Current Status and Future Directions”. Wirel. Pers. Commun,97(3), 2017,pp.3355–3425

[3] H. Alzaid, E. Foo, J.G. Nieto, “Secure data aggregation in wireless sensor network: a survey”, in: Proc. of the Australasian Information Security Conference, 2008, pp. 93–106.

[4] Guo, J., Fang, J. a. and Chen, X., "Survey on secure data aggregation for wireless sensor networks" ,in Service Operations, Logistics,and Informatics(SOLI),IEEE InternationalConference on, 2011,pp.138-143.

[5] Yubo Wang, Liang Li, Chen Ao, Puning Zhang, Zheng Wang, and Xinyang Zhao,“Secure Data Aggregation Mechanism based on Constrained Supervision for Wireless Sensor Network”, Intl Journal of Electronics and Telecommunications(IJET), Vol. 65, No. 2,2019, pp. 259–266.

[6] Priyanka B. Gaikwad, Manisha R. Dhage, “ Survey on Secure Data Aggregation in Wireless Sensor Networks, Computing Communication Control and Automation (ICCUBEA)”, International Conference ,2015, pp. 242-246

[7] Mohammad Youssef, Raghav Yadav, “Survey on Several Secure Data Aggregation Schemes in WSN”, International Journal of Current Engineering and Technology, Vol.6, No.4, 2016. pp. 1154-1159

[8] Atif Alamri, et al., A survey on sensor-cloud: architecture, 2230 applications, and approaches,2013, Int. J. Distrib. Sens. Netw. 20132231.

[9] Sang, Y.P.; Shen, H.; Inoguchi, Y.; Tan, Y.; Xiong, N. “Secure data aggregation in wireless sensor networks: A survey”. In Proceedings of the 7th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT’06), Taipei, China, 4–7 December 2006; pp. 315–320.

[10] Ozdemir, S.; Yang, X. “Integrity protecting hierarchical concealed data aggregation for wireless sensor networks”. Comput. Netw. 2011, 55, pp.1735–1746.

[11] Granjal, J., Monteiro, E., & Silva, J. S. “Security in the integration of low-power Wireless Sensor Networks with the Internet: A survey”. Ad Hoc Networks, 24, 2015, pp. 264–287.

[12] H. Hayouni and M. Hamdi, "Secure data aggregation with homomorphic primitives in wireless sensor networks: A critical survey and open research issues," 2016 IEEE 13th International Conference on Networking, Sensing, and Control (ICNSC), Mexico City, 2016, pp. 1-6.

[13] Sathya, Duraisamy & Pugalendhi, Ganeshkumar.”Secured data aggregation in wireless sensor networks”. Sensor Review. March 2018, pp. 1-7.

[14] Vinodha, D., & Mary Anita, E. A. “Secure Data Aggregation Techniques for Wireless Sensor Networks: A Review.” Archives of Computational Methods in Engineering. Springer, September 2019, Volume 26, Issue 4, pp. 1007–1027

[15] X. Liu, J. Yu, F. Li, W. Lv, Y.Wang and X. Cheng,"Data Aggregation in Wireless Sensor Networks:From the Perspective of Security,"in IEEE Internet of Things Journal ,2109, pp.1-2, doi:10.1109/JIOT.2019.2957396,

[16] Westhoff, D.; Girao, J.; Acharya, M. “Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption keydistribution and routing adaptation”. IEEE Trans. Mobile Comput 2006,5, pp.1417–1431.

[17] Niu, S.F.; Wang, C.F.; Yu, Z.X.; Cao, S. “Lossy data aggregation integrity scheme in wireless sensor networks”. Comput. Electr. Eng. 2013, 39, pp.1726–1735

[18] Zhou, Q.; Yang, G.; He, L.W. “An efficient secure data aggregation based on homomorphic primitives in wireless sensor networks”. Int. J. Distrib. Sens. Netw. 2014, 962925.

[19] J. Domingo-Ferrer, “A Provably Secure Additive and Multiplicative Privacy Homomorphism,” Proc. Information Security Conf.,2002, pp. 471-483.

[20]] Y. Yun, Y. Qian and H. Sharif, "A Secure Data Aggregation and Dispatch Scheme for Home Area Networks in Smart Grid," IEEE Global Telecommunications Conference - GLOBECOM 2011, Houston, TX, USA, 2011, pp. 1-6.

[21] N. Alamatsaz, A. Boustani, M. Jadliwala and V. Namboodiri, "AgSec: Secure and efficient CDMA-based aggregation for smart metering systems,"IEEE 11th Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, 2014, pp. 489-494.

[22] Abdallah, A., & Shen, X. S. “ A lightweight lattice-based homomorphic privacy-preserving data aggregation scheme for smart grid.” IEEE Transactions on Smart Grid, 9(1),2018,pp. 396-405

[23] R. B. Romdhane, H. Hammami, M. Hamdi and T. Kim, "At the cross roads of lattice-based and homomorphic encryption to secure data aggregation in smart grid," 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 2019, pp. 1067-1072.

[24] Ali Broumandnia, “Modular Matrix Inverse in Zn,” https://www.mathworks.com › matlabcentral › 64813-modular-matrix-inverse in Zn.

[25] Lecture 33: Left and right inverses; pseudoinverse – MIT. https://ocw.mit.edu › courses › positive-definite-matrices-and-applications

[26] James, M. (June 1978), “The generalised inverse,”. Mathematical Gazette. 62 (420): pp. 109–114.

[27] “Null Space and Nullity of a Matrix” – GeeksforGeeks. https://www.geeksforgeeks.org › null-space-and-nullity-of-a- matrix.

[28] ASA314 – “Matrix Inversion with Modulo Arithmetic” – People https://people.sc.fsu.edu › ~jburkardt › cpp_src › asa314.

[29] ES.1803 Topic 14 Notes 14 Row reduction and subspaces – MIT, web.mit.edu › jorloff › www › 18.03-esg › notes ›topic14

[30] Lyubashevsky, V and Micciancio, D. “Asymptotically Efficient Lattice-Based Digital Signatures”.2018, Journal of Cryptology 31, pp. 774–797.

[31] Gupta, D. S., & Biswas, G. P, “Design of lattice-based ElGamal encryption and signature schemes using SIS problem,” Transactions on Emerging Telecommunications Technologies, 2017,29(6), pp.1-20.