PLOS ONE

RESEARCH ARTICLE

Cost-Efficient and Multi-Functional Secure Aggregation in Large Scale Distributed Application

Ping Zhang[1]*, **Wenjun Li**[2,3], **Hua Sun**[4]*

1 School of Electronics and Information Engineering, Hunan University of Science and Engineering, Hunan, P.R. China, **2** Hunan Provincial Key Laboratory of Network Investigational Technology, Hunan Police Academy, Hunan, P.R.China, **3** Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, Changsha University of Science and Technology, Hunan, P.R. China, **4** School of Information Engineering, Changsha Medical University, Hunan, P.R. China

* pingzp@hotmail.com (PZ); sunhua0418@126.com (HS)

OPEN ACCESS

Citation: Zhang P, Li W, Sun H (2016) Cost-Efficient and Multi-Functional Secure Aggregation in Large Scale Distributed Application. PLoS ONE 11(8): e0159605. doi:10.1371/journal.pone.0159605

Editor: Francesco Pappalardo, Universita degli Studi di Catania, ITALY

Received: January 13, 2016

Accepted: July 5, 2016

Published: August 23, 2016

Data Availability Statement: All relevant data files are available from the database of Tropical Atmosphere Ocean (TAO) project (http://www.pmel.noaa.gov/tao).

Funding: This work is supported by the Foundation of Hunan Educational Committee (14C0484), National Natural Science Foundation of China under Grant (61502054), Yongzhou Science and Technology Plan ([2013]3), Open Research Fund of Hunan Provincial Key Laboratory of Network Investigational Technology (2016WLZC016), and Foundation of Hunan University of Science and Engineering (13XKYTA003).

Abstract

Secure aggregation is an essential component of modern distributed applications and data mining platforms. Aggregated statistical results are typically adopted in constructing a data cube for data analysis at multiple abstraction levels in data warehouse platforms. Generating different types of statistical results efficiently at the same time (or referred to as enabling multi-functional support) is a fundamental requirement in practice. However, most of the existing schemes support a very limited number of statistics. Securely obtaining typical statistical results simultaneously in the distribution system, without recovering the original data, is still an open problem. In this paper, we present SEDAR, which is a SEcure Data Aggregation scheme under the Range segmentation model. Range segmentation model is proposed to reduce the communication cost by capturing the data characteristics, and different range uses different aggregation strategy. For raw data in the dominant range, SEDAR encodes them into well defined vectors to provide value-preservation and order-preservation, and thus provides the basis for multi-functional aggregation. A homomorphic encryption scheme is used to achieve data privacy. We also present two enhanced versions. The first one is a Random based SEDAR (REDAR), and the second is a Compression based SEDAR (CEDAR). Both of them can significantly reduce communication cost with the trade-off lower security and lower accuracy, respectively. Experimental evaluations, based on six different scenes of real data, show that all of them have an excellent performance on cost and accuracy.

Introduction

Enormous amounts of rich diverse information are constantly generated in modern large distributed systems, which are also called big data. Such large-scale big data sources create exciting opportunities for service quality monitoring, novelty discovery, or attack detection, etc.

Competing Interests: The authors have declared that no competing interests exist.

However, directly transmitting them to a single node and processing using centralized algorithms are difficult. Distributed aggregation is an efficient way to minimize consumption of energy and bandwidth.

Typical distributed application scenarios can be easy to found big internet firms, such as Google and Bing. Click log data of these service providers is distributed on thousands of servers around the world, which is usually up to megabytes per minute. In these distributed big data scenarios, 90% of regular analytics jobs issue queries against different type of aggregated values, instead of requiring the raw log data records. To efficient generate these aggregated results, performance metrics is generated locally from log data, and then a distributed aggregation can be adopted [1].

As another example, consider the WSNs application scenarios. In these applications, nodes are often equipped with a battery as the energy unit, which means the energy capacities are limited. Meanwhile, such WSNs are envisioned to be spread out over a large geographical area, and the total number of the nodes is huge, so the battery change is impossible. How to save the overall energy resources and extend the lifetime of the networks is essential. Distributed aggregation is also a popular research topic in this area [2].

Enabling multi-functional support is a fundamental requirement in practice. Here, multi-functional support means to provide as many statistical results as possible. Typical aggregation functions include count, summation, mean, median, maximum, minimum, variance, mode, etc. These statistical results are typically adopted in constructing a data cube for data analysis at multiple abstraction levels in data warehouse platforms [3], In order to improve the performance of data mining, it is a basic requirement to keep data feature (i.e. statistics) as much as possible in data cubes, which means that enabling multi-functional support is necessary for corresponding distributed aggregation schemes. System wide properties generated from data aggregation, can also be used as input parameters for other distributed applications and algorithms, or utilized for decision making directly. For example, setting the fan-out of gossip protocol [4] in peer-to-peer applications, or achieving load balancing in content delivery networks [5] need aggregation results as their parameters or inputs.

Serval distributed aggregation schemes [6–8] have been put forward. Security is a basic requirement for most applications. Serval secure distributed aggregation schemes [2, 9–13] have likewise existed. However, most of them can only achieve a very limited type of statistics, and even combine several existed schemes still can't respond to the request. In fact, efficient obtaining global-related statistical results, such as median and mode, in a distributed manner, even without considering security problem, is still a challenge [3].

In RCDA [14], a homomorphic encryption algorithm is used to provide end-to-end confidentiality, and simple concatenation all sensing data without any information compression method to enable recoverability of all sensing data. Although the scheme can achieve arbitrary method support, the communication cost is too heavy to be applied to large scale networks. Based on RCDA, EERCDA [13] uses a differential data transfer method to reduce the communication cost, in which the difference data rather than raw data from the sensor node are transmitted to the cluster head. However, the total transmission overhead still too heavy for most time.

To the best of our knowledge, securely obtaining typical statistical results simultaneously in the distribution system, without recovering the original data, is still an open problem.

In this paper, we study the problem of *multi-functional secure distributed aggregation*, in which all the aggregation functions mentioned above can be obtained securely in a single aggregation query. We also propose three complementary schemes to work around this problem. We first present SEDAR, which is a SEcure Data Aggregation scheme under the Range segmentation model, and then proposed two enhanced version REDAR (Random based SEDAR) and CEDAR (Compression based SEDAR).

To reduce the communication cost by capturing the data characteristics, a range segmentation model is adopted in proposed schemes, and different range uses different aggregation strategy. Raw data in dominant range are encoded into well defined vectors at each node to preserve both the order-related and the value-related information during distributed aggregation, and thus different types of statistics can be obtained simultaneously recovering the original data. The vectors are encrypted by a homomorphic scheme, and encrypted vectors are aggregated directly in cipher domain at an intermediate node, so concealment is also achieved. Raw data in other range will be encrypted by traditional asymmetric encryption schemes, and transmitted without in-network aggregation.

The major contributions of this paper are summarised as follows.

- We propose a novel and practical scheme, called SEDAR, in which all common statistical results can be securely and efficiently obtained without recovering the original data.
- We also present two enhanced versions, namely REDAR and CEDAR, to further reduce the communication cost with the trade-off lower security and lower accuracy, respectively.
- We implement these three schemes and extensively evaluate their performance. Evaluation results, based on six different scenes of real data, show that all of them have an excellent performance on cost and accuracy.

The remaining parts of this paper are structured as follows. Section 2 describes terminologies, and additional background knowledge. Section 3, 4 and 5 introduce SEDAR, REDAR and CEDAR. Section 6 and 7 is performance analysis and evaluation results. Section 8 briefly examines the related work. Section 9 provides a summary.

Preliminaries

In this section, we first give a range segmentation model and a network model, and then present problem definition. We also introduce a homomorphic encryption scheme.

Range Segmentation Model

An illustration of range segmentation model is given in Fig 1, terminologies used in this model are defined as follows.

Definition 1 *(R_m, measurement range) Measurement ranges are those over which the measurement instruments are calibrated. Convincing and reliable results of a given instrument will only appear in its measurement range, i.e., $R_m = [X_{LM}, X_{UM}]$, s.t. $R_m \subseteq R$*

Definition 2 *(R_e, effective range, operation range, valid range) Effective range is the set of allowed values for a variable in a concrete application. It's a subset of R_m, i.e., $R_e = [X_{LE}, X_{UE}]$, s.t. $R_e \subseteq R_m$*

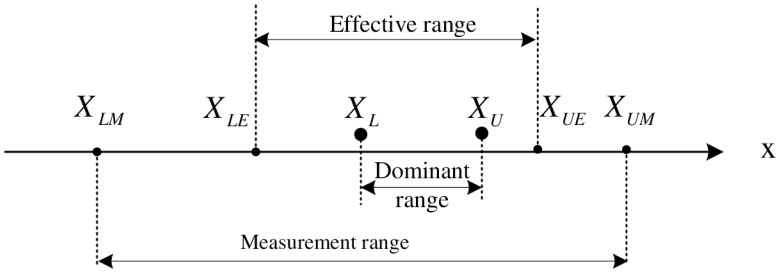

Fig 1. Range Segmentation Model.

doi:10.1371/journal.pone.0159605.g001

Definition 3 *(R_d, dominant range, dominant area, advantaged region, main range) Dominant range is a subset of R_e, whose probability is significantly greater than other's, i.e.,* $R_d = [X_L, X_U]$ *s.t.* $R_d \subseteq R_e$ && $P(R_e) - P(R_d) \ll 1$

Definition 4 *(R_b, border region, boundary region, margin area) Border region is the set of allowed values that outside the dominant range. It's a subset of R_e, i.e.,* $R_b = [X_{LE}, X_{UE}] = \overline{R_d} R_e$ *s.t.* $R_b \cap R_d = \emptyset$ && $R_b \cup R_d = R_e$

Network Model

The network is modeled as a connected graph $G = (V, E)$, with $|V|$ vertices and $|E|$ links. Each vertex represents a network node and each link represents a communication channel. Node is a logical concept. For example, in global scale distribution systems, each data center can be regarded as a node.

The sink node $S \in V$, which has a powerful computing and storage capacity, is a trusted node. S is also known as query server. The remainder nodes $C \subset V$ are either reliable or unreliable, each node only has one parent node. $|C| = N$. $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$ is raw data generated at these nodes. A set of nodes A are selected as aggregator nodes, $A \subset C$. The aggregator nodes are also performed as cluster heads, the others nodes ($C\overline{A}$) are cluster members. Each cluster member join an appropriate cluster according certain criterion, such as signal strength in wireless network and delay in wired network.

Problem Definition

Definition 5 *(Data aggregation) Given a dataset $X = \{x_1, x_2, \ldots, x_N\}$, a aggregation function set $F = \{f, h, \ldots\}$, and a aggregation result set $Y = \{y_1, y_2, \ldots\}$, a data aggregation is defined as $Y = F(X)$, s.t. $|Y| \ll |X|$.*

Definition 6 *(Distributed Aggregation) Given a network G, X is the raw data generated at each node, divide X into several subsets $\{X_1, X_2, X_M\}$,* s.t. $M < N, X_1 \cup X_2 \ldots \cup X_M = X, X_1 \cap X_2 \ldots \cap X_M = \phi$. *An in-network data aggregation is defined as* $y = F(X) = \mathrm{f}(\mathrm{h}(X_1), \ldots, \mathrm{h}(X_M))$. *Each subset can be further divided, and this definition is still satisfied.*

Data aggregation of each subset is accomplished at aggregator nodes, and the final data aggregation is executed at the query server.

Definition 7 *(MFSDA, Multifunction Secure Distributed Aggregation) An MFSDA is a distributed aggregation which can provide both privacy confidentiality and multi-functional supporting. Multi-functional means that several statistical results can obtain efficiently in the same query, and results include at least count, summation, average, median, maximum, minimum, variance and standard deviation.*

Homomorphic Encryption Scheme

The traditional encryption technology is not suitable for secure distributed aggregation. It only provides concealment but do not support cipher text operations, so the intermediate aggregators will have to decrypt the received data before aggregation. And then the aggregated results need be re-encrypted before sending. Frequent encryption and decryption in intermediate nodes will increase the computing cost and the energy consumption. The key management is also difficult. Each intermediate node has to maintain the private key for decryption, which will increase the risk of leaks.

To reduce the computing cost and enhance the security, a homomorphic encryption scheme is used in the proposed schemes. It is derived from homomorphism in the abstract algebra. By using homomorphism, operations in one algebraic system (plaintext) can be

mapped into operation in another algebraic system (cipher text), which means data aggregation can perform on cipher text directly, and only the sink node needs to store the private for decryption. Homomorphic encryption technology includes partially homomorphic and fully homomorphic. In theory, based on the fully homomorphic, all these statistics can be easily computed. However, the fully homomorphic encryption, while revolutionary, is not really practical. Practitioners rely therefore on already existing partially homomorphic encryption, which is constructed from traditional encryption schemes and has been widely used in multi-party computation, electronic voting, non-interactive verifiable secret sharing, e-auction, and others [15, 16].

Homomorphic encryption used in proposed schemes is a partially homomorphic which can only allow homomorphic computation of only one operation (i.e., addition). To further reduce key size with high security and benefit us with the computation cost, an ElGamal Encryption Scheme (EC-EG) [2, 14], which is an elliptic curve based encryption, will be used here. EC-EG is also an asymmetric homomorphic encryption scheme, so the key management is easy. It consists of four parts: Setup, KeyGen, Encryption (HEnc) and Decryption (HDec). Its ciphertext is an elliptic curve over a finite field, and $\oplus$ is the point addition on elliptic curves.

Theorem 1 (Additive Homomorphic Encryption) *EC-EG is an additive homomorphic encryption scheme, namely the addition in plaintext is equivalent to point addition in cipher domain, i.e.,* $m_1 + m_2 = HDec(HEnc(m_1) \oplus HEnc(m_2))$

SEDAR

In this section, we introduce SEDAR to solve the *multi functional secure distribution aggregation* problem. We first give a brief overview of SEDAR. Then, a detail version is presented. Finally, a concrete example is given.

Overview

In the proposed scheme, aggregation is performed in cipher domain. Both the sub-aggregation and the final aggregation results are encrypted vectors, which can be decrypted using the private key owned by the server. All statistics are calculated directly from the final aggregated vector at the server, which makes thing much easier.

As shown in Fig 2, there are five steps in the proposed scheme: mapping, encoding, encryption, aggregation, and decryption. The first three are executed on each node independently, while the last two are executed at aggregation nodes and the server respectively. Mapping and encoding are used to enable multi-function supporting, while the other three are used to achieve data confidentiality.

There are also five kinds of data corresponding to each step: raw data x, mapped data y, encoded data $\vec{v}$, encrypted data $\vec{c}$, aggregation of encrypted data $\vec{C}$, and aggregation of encoded data $\vec{V}$.

Raw data x_k is the original data gathered at node k, which is belong to a subset of real domain. This real domain is split into several partition based on the range segmentation model, and different strategies will choice for different range.

The lower bound of R_d is defined as $X_L = max(X_{LE}, \hat{\mu} - \beta\hat{\delta})$, and the upper one is $X_U = min(X_{UE}, \hat{\mu} + \beta\hat{\delta})$. $\vec{\mu}$ and $\vec{\delta}$ are mean and standard deviation estimated using historical or empirical data. β is a factor. $\beta \in [1.8, 3]$, and $\beta = 2$ satisfies most application. The bounds of R_e and R_m are determined by the application itself.

Effective data in the dominant range ($x_k \in R_d$) will be transformed into *mapped data* y_k using the *mapping* function. y_k is belong to a subset of the natural numbers, i.e. $y_k \in (0, L]$,

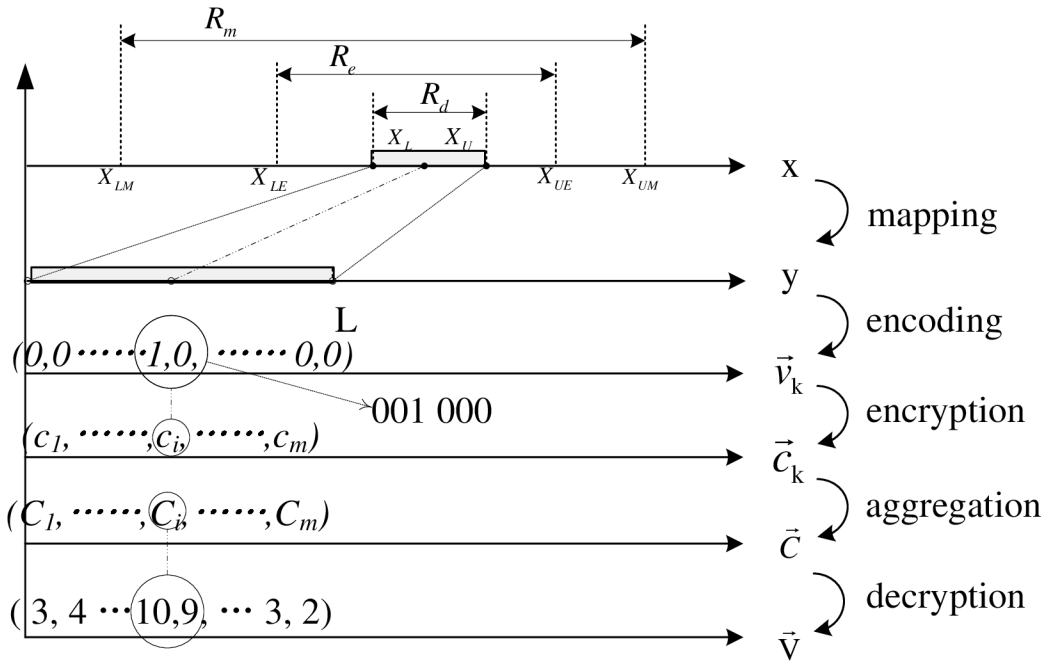

Fig 2. SEDAR.

doi:10.1371/journal.pone.0159605.g002

where $L = \lceil \frac{X_U - X_L}{a} \rceil$ and a is the accuracy requirement of x_k. The conversion between x_k and y_k is achieve by the *mapping* function and its inverse, i.e. f_m and f_m^{-1}. To achieve value-preserved and order-preserved during this conversion, f_m and f_m^{-1} should be monotonic functions. The mapping function can be a linear one or a nonlinear one, which can be either a piecewise function or a non piecewise function. For example, a linear *mapping* function and its inverse are like $y_k = f_m(x_k) = \frac{x_k - X_{LB}}{a}$, and $x_k = f_m^{-1}(y_k) = a \times y_k + X_{LB}$.

Effective data outside the dominant range ($x_k \in R_b$) will be encrypted by traditional asymmetric encryption scheme, and transmitted without in-network aggregation. As $P(R_e) - P(R_d) \ll 1$, serial transmission of these data will not increase the transmission overhead significantly. Abnormal data that outside R_e are also reported to the server without aggregation.

Encoded data $\vec{v}_k$ is a vector, $\vec{v}_k \in \{0, 1\}^L$, where L is the number of elements. The $(y_k)th$ element of $\vec{v}_k$ is 1, and all other elements is 0. The conversion between y_k and $\vec{v}_k$ is achieve by the *encoding* function and its inverse, i.e., $\vec{v}_k = f_e(y_k)$, and $y_k = f_e^{-1}(\vec{v}_k)$. For example, the encoding function can be programmed as two instructions, i.e. $\vec{v}_k = zeros(1, L)$ and $\vec{v}_k(y_k) = 1$, while its inverse function can be achieved by $y_k = find(\vec{v}_k > 0)$.

Each node k encrypts its $\vec{v}_k$ into $\vec{c}_k$ using the homomorphic encryption scheme, and sent $\vec{c}_k$ to its parent. As all encrypted data $\vec{c}_k$ are generated with the same public key, we can aggregation it directly in ciphertext $\vec{C} = \oplus \vec{c}_k$. According homomorphism, the final aggregation result $\vec{V}$ can be obtain by decrypting $\vec{C}$. Then the typical statistical results can be obtained from $\vec{V}$.

Detail of SEDAR

SEDAR consists of three procedures: *Setup*, *Operations on Clients*, and *Operations on Server*.

Setup. The *Setup* procedure performs network initialization, encryption initialization, and parameters distribution.

Boundary definitions (i.e., R_e, R_d, and R_b) and accuracy requirement (i.e. a) are distributed into each node, include clients and server. Itm_{size}, B_{msize}, B_{csize}, and B_{num} mentioned above are also public information.

There are two encryption schemes used in this paper: a traditional one and a homomorphic one. Both of them are public key cryptogram schemes.

The traditional encryption scheme is used for R_b, where the key pair is {*KPriS*, *KPubS*}, the encryption function is $C = Enc(msg, KPubS)$, and the decryption function is $msg = Dec(C, KPriS)$.

The homomorphic encryption scheme is used for R_d, where the key pair is {*KPriSH*, *KPubSH*}, the encryption function is $C = HEnc(msg, KPubSH)$, and the decryption function is $msg = HDec(C, KPriSH)$.

Both *KPubS* and *KPubSH* are public information, while private keys (i.e. *KPriS* and *KPriSH*) must be keep in privacy only by the server.

Operations on Clients. It consists of three parts: *local data processing*, *received data processing* and *data transmission*.

As shown in algorithm 1, in *local data processing*, each node gathers the raw data x_i, and process it according the range definition. The traditional encryption scheme will be used for the data belong to R_b, the encrypted data will add into $bSet_i$. For data in R_d, *mapping* f_m and *encoding* f_e function will be used before the homomorphic encryption, the encrypted data will add into $hSet_i$. For the data outside the valid range R_e, the node ID will be added into the alarm set $aSet_i$ after being encrypted.

Algorithm 1 Operations on Client (Part I)

```
1: procedure LOCAL DATA PROCESSING
2:   if (x_i ∈ R_b) then
3:     C_i ← Enc(x_i, KPubS)
4:     bSet_i ← {C_i}
5:   else if (x_i ∈ R_d) then
6:     y_i ← f_m(x_i)                                   ▷ Mapping
7:     v_i ← f_e(y_i),  s.t. v_i ∈ {0,1}^L.             ▷ Encoding
8:     C_hi ← nul
9:     for j ← 0 to B_num − 1 do
10:      start ← max(1, (|v|_i(−(j + 1)B_msize + 1)
11:      end ← |v_i| − jB_msize
12:      m ← v_i[start, end]
13:      c ← HEnc(m, KeyPubSH)
14:      C_hi ← [c, C_hi]
15:    end for
16:    hSet_i ← {C_hi}
17:  else
18:    CID_i ← Enc(ID_i, KPubS)
19:    aSet_i ← {CID_i}
20:  end if
21: end procedure
```

As shown in algorithm 2, the *received data processing* only exists in cluster header (i.e., CHs). Each CH use it to deal with packets received from its children (i.e., CMs). Items in each packet will be classified into three sets, i.e., $bSet_i$, $hSet_i$, and $aSet_i$. All items of $hSet_i$ will be aggregated directly in cipher domain, and the aggregation result is Ch_i.

In the *data transmission*, all processing results Ch_i, $bSet_i$, and $aSet_i$ are send to its parent.

Each element in $\vec{v}_k$ (i.e., $v_k(i)$) or $\vec{V}$ is allocated the same size (denote as, Itm_{size} or $|v_k(i)|$), it is influenced by N and the distribution of x. $Itm_{size} \in \left(\lceil log\frac{N}{L}\rceil, \lceil logN\rceil\right)$.

The maximum size (denote as P_{size}) that the homomorphic encryption function can operate each time, is always large than Itm_{size}. To reduce the total ciphertext size and the computation cost, several adjacent vector elements can be encrypted at the same time. For example, in Fig 2, each element is allocated three bits, and every two elements are encrypted with each other.

The maximum number of vector elements that can be encrypted by the encryption function is $\lfloor \frac{P_{size}}{Itm_{size}} \rfloor$, and the actual size of plaintext for the encryption function is $B_{msize} = \lfloor \frac{P_{size}}{Itm_{size}} \rfloor Itm_{size}$. The corresponding ciphertext size is denoted as B_{csize}. When $|\vec{v}_k|$ is much larger than B_{msize}, the homomorphic encryption function need to repeat $B_{num} = \lceil \frac{|\vec{v}_k|}{B_{msize}} \rceil$ times to finish the encryption for $\vec{v}_k$. The decryption function also needs to repeat B_{num} times.

Algorithm 2 Operations on Client (Part II)

```
1: procedure RECEIVED DATA PROCESSING
2:   if current node is a CH then
3:     for all received Packet_k do
4:       extract {C_hk, bSet_k, aSet_k} from Packet_k
5:       bSet_i ← bSet_i ∪ bSet_k
6:       aSet_i ← aSet_i ∪ aSet_k
7:       hSet_i ← hSet_i ∪ {C_hk}
8:     end for
9:     for j ← 1 to B_num do
10:      ct_1 is initialized as the infinity point of E
11:      for all C_hk ∈ hSet do
12:        ct_2 ← C_hk[(j − 1) B_csize + 1, jB_csize]
13:        ct_1 ← ct_1 ⊕ ct_2
14:      end for
15:      C_hi[(j − 1) B_csize + 1, jB_csize] ← ct_1
16:    end for
17:  end if
18: end procedure
```

Operations on Sever. Operations on *server* consist of five parts: *data receiving and retrieving*, *boundary range data processing*, *alarm data processing*, *dominant range data processing*, and *obtain statistical results*. The first three are contained in algorithm 3, while the others are contained in algorithm 4 and 5.

Algorithm 3 Operations on Server (Part I)

```
1: procedure OPERATIONS ON SERVER
2:   for all received Packet_k do                    ▷ data retrieving
3:     extract {C_hk, bSet_k, aSet_k} from Packet_k
4:     bSet ← ⋃ bSet_k
5:     aSet ← ⋃ aSet_k
6:     hSet ← ⋃ C_hk
7:   end for
8:   mSet ← { }                                     ▷ boundary range data processing
9:   for all C_i ∈ bSet do
10:    m_i ← Dec(KPriS, C_i)
11:    mSet ← ⋃ {m_i}
12:  end for
13:  for all cid ∈ aSet do                          ▷ alarm data processing
14:    ID_i ← Dec(KPriS, cid)
15:    treat ID_i as a potential abnormal node
16:  end for
17: end procedure
```

In the *data receiving and retrieving*, the server receives all packets from its children. All its children are CH, and the packets like $\{C_{hi}, bSet_i, aSet_i\}$. Items in each packet will be classified into three sets, i.e. *bSet*, *aSet* and *hSet*.

Items in *bSet* are boundary range data, all of them are encrypted in traditional scheme. The decrypted data are added into *mSet*.

Items in *aSet* are IDs of clients which data is out of boundary range. Those nodes are treated as potential abnormal node, and may need further analysis.

Items in *hSet* are dominant range data, all of them are homomorphic encrypted data, so all items in this set can be aggregated directly in cipher domain. After decrypting the aggregated encrypted data using *KPriSH*, we get an aggregation result of data in dominant range, in a vector form, i.e., $\vec{V} = \{n_1, n_2, \ldots, n_L\}$.

Algorithm 4 Operations on Server (Part II)

```
1: procedure OPERATIONS ON SERVER
                                          ▷ dominant range data processing
2:   for j ← 1 to B_num do
3:     ct_1 is initialized as the infinity point of E
4:     for all C_i ∈ hSet do
5:       ct_2 ← C_i[ (j − 1) B_csize + 1, jB_csize]
6:       ct_1 ← ct_1 ⊕ ct_2
7:     end for
8:     tmp ← HDec(t_1, keyPriS)
9:     M⃗ ← [M⃗, tmp]
10:  end for
11:  V⃗ ← nul
12:  B_size ← |M| / L
13:  for j ← 1 to L do
14:    n_j ← M[ |M| − j Itm_size + 1, |M| − (j − 1) Itm_size]
15:    V⃗ ← [n_j, V⃗]
16:  end for
17: end procedure
```

Finally, each statistical results can be obtained directly from $\vec{V}$ and *mSet* by algorithm 5.

Property of SEDAR

Multi-function. On the one hand, the value-related information needs to be preserved in the transformation for the summation based statistics. y_k can be recovered from $\vec{V}(i)$, and x_k can be recovered from y_k. $\vec{V}(i)$ itself represents how many $x_k = f_m^{-1}(i)$ in the raw data. So the value-related information is maintained in the $\vec{V}$.

On the other hand, the order-related information needs to be preserved in the transformation for the comparison based statistics. Assuming that $\vec{V}(i) \neq 0$, $\vec{V}(j) \neq 0$, and $i > j$. We recover the raw data as $x_i = f_m^{-1}(i)$ and $x_j = f_m^{-1}(j)$, and then we can use the monotonicity of f_m to judge which one is larger. So order-related information is maintained in the vector $\vec{V}$.

Therefore, both the summation based statistics and the comparison based statistics can be calculated in the proposed scheme.

Data Privacy. On the one hand, the adversary cannot infer the true position of the non-zero element from an encrypted vector, and thus can not recover x_k by using these information. There is a random function in the homomorphic encryption scheme. Even if two elements have the same value, their ciphertexts are still different from each other. For example, in the leaf nodes, each encoded vectors contain only a non-zero elements, and all other

elements are zero. It's easily infering the corresponding value x_k in plaintext domain by obtaining the position i of the non-zero elements and using $f_m^{-1}f_e^{-1}(i)$. However, in ciphertext domain, encrypted elements are different to each other, and even encrypted zero elements are also different to each other. As a result, inferring the position of non-zero elements is difficult in encrypted vectors.

Algorithm 5 Operations on Server (Part III)

procedure OBTAIN STATISTIC RESULT
  CNT $\leftarrow \sum_{i=1}^{L} n_i + mSet$ ▷ Count
  SUM $\leftarrow \sum_{i=1}^{L} f_m^{-1}(i) \times n_i + \sum_{mSet} x_i$ ▷ Summation
  MEAN $\leftarrow \frac{SUM}{CNT}$ ▷ Average/mean
  $i_{max} \leftarrow max(\{\ i\ i \in (0, L]\ \&\&n_i > 0\}\ )$
  $max_1 \leftarrow f_m^{-1}(i_{max})$
  $max_2 \leftarrow max(mSet)$
  MAX $\leftarrow max\{max_1, max_2\}$ ▷ Maximum
  $i_{min} \leftarrow min(\{\ i\ i \in (0, L]\ \&\&n_i > 0\}\ )$
  $min_1 \leftarrow f_m^{-1}(i_{min})$
  $min_2 \leftarrow min(mSet)$
  MIN $\leftarrow min\ min_1, min_2$ ▷ Minimum
  $cnt_L \leftarrow \#\{\ mSet(^* < MIN_1)\}$,
    ▷ cnt_L is the total number in $\{mSet\}$ whose value is less then or equal to MIN.
  $M \leftarrow min\left(\{j|\sum_{i=1}^{j} n_i \geq \lceil\frac{CNT}{2}\rceil - cnt_L\}\right)$
  $M' \leftarrow min\left(\{j|\sum_{i=1}^{j} n_i \geq \lceil\frac{CNT+1}{2}\rceil - cnt_L\}\right)$
  if CNT is odd **then** ▷ Median
    MEDIAN $\leftarrow f_m^{-1}(M)$
  else
    MEDIAN $\leftarrow \frac{f_m^{-1}(M)+f_m^{-1}(M')}{2}$
  end if
  $SUM(\hat{x}^2) \leftarrow \sum_{i=1}^{L} (f_m^{-1}(i))^2 n_i + \sum_{mSet} {x_i}^2$
  $E(\hat{x}^2) \leftarrow \frac{SUM(\hat{x}^2)}{CNT}$
  $E(\hat{x})^2 \leftarrow MEAN^2$
  VAR $\leftarrow E(x^2) - E(x)^2$ ▷ Variance
  STD $\leftarrow \sqrt{VAR}$ ▷ Standard deviation
  $m1cnt \leftarrow max(\vec{V})$
  $i_{mode1} \leftarrow find(\vec{V} == m1cnt)$
  $mode2 \leftarrow mode(mSet)$
  $m2cnt \leftarrow sum(mSet == mode2)$
  if $m1cnt > m2cnt$ **then** ▷ Mode
    MODE $\leftarrow f_m^{-1}(i_{mode1})$
  else
    MODE $\leftarrow mode2$
  end if
end procedure

On the other hand, all message relayed or aggregated in the intermediate node are encrypted message. Due to the homomorphic encryption scheme, the aggregation for data in R_d performs on cipher text of $\vec{v}_k$ directly. Message in *aSet* and *bSet* are encrypted by a traditional encryption scheme. So all message relayed or aggregated in the intermediate node are encrypted message, all private keys keep in privacy only by the server. Without private key, no one can decrypt, so data confidentiality is achieved.

Table 1. Example for SEDAR.

ID	Raw data	Range	Result
1	32	R_d	(0 1 0 0)
2	16	$\overline{R}_e$	*aSet* = {"3"}
3	32	R_d	(0 1 0 0)
4	33	R_d	(0 0 1 0)
5	28	$R_e\overline{R}_d$	*mSet* = {28}
6	33	R_d	(0 0 1 0)
7	34	R_d	(0 0 0 1)
8	49	$\overline{R}_e$	*aSet* = {"8"}
9	33	R_d	(0 0 1 0)
10	25	$R_e\overline{R}_d$	*mSet* = {25}

doi:10.1371/journal.pone.0159605.t001

A Concrete Example for SEDAR

Assuming $R_e = (20, 40]$, $R_d = (30, 34]$ and accuracy requirement is $a = 1$. As shown in Table 1, there are 10 nodes in the given network. Raw data of each node list in the 2nd column, the 3rd and 4th columns are data classification and processing results.

Raw data of node 2 and node 8 are outside of the valid data range $R_e = (20, 40]$. Both of them will be regarded as illegal data and discarded, and their IDs will be added into alarm set *aSet*.

Raw data of node 5 and node 10 are in the boundary range $R_b = R_e\overline{R}_d = (20, 30] \bigcup (34, 40]$. Both of them will be added into boundary set *bSet*.

Other raw data are in the dominant range R_d. Each of them will be transformed into y_k ($y_k \in (0, 4]$) by using the *mapping function*. Each valid mapped data y_k will then be encoded into a vector $\vec{v}_k$ whose length is L. The y_k-th elements is 1, while all the remaining elements are set to 0. For example, in node 1, the raw data is $x_1 = 32$, the mapped data $y_k = 2$ is obtained after mapping step, and in the encoding step, the 2nd (y_k-th) element of the vector is set to 1, while other elements are 0, i.e. $\vec{v}_k = (0\ 1\ 0\ 0)$.

Each vector will be encrypted by the homomorphic encryption scheme and in-network aggregation will perform directly in cipher domain.

Elements of *aSet* and *bSet* will be encrypted by the traditional encryption scheme, and be relayed to the server without in-network aggregation.

According the homomorphic property, the aggregation of vectors in cipher text domain is equivalent to that in plaintext. Therefore, the *server* can obtain the final aggregation result $\vec{V} = \sum \vec{v}_k$ by decrypting the received data. Encrypted data in *aSet* and *bSet* can also be decrypted by server. The final data obtained at the server include

$$\begin{aligned} \vec{V} &= \sum \vec{v}_k = (0\ 2\ 3\ 1) \\ mSet &= \bigcup mSet_i = \{25, 28\} \\ aSet &= \bigcup aSet_i = \{\text{“3”}, \text{“8”}\} \end{aligned}$$

Each statistic can be calculated using algorithm 5.

CNT = (2 + 3 + 1) + 2 = 8;

SUM = 2 × (2 + 30) + 3 × (3 + 30) + 1 × (4 + 30) + (25 + 28) = 250;

MEAN = 31.25;

$i_{max} = 4$, $max_1 = 34$, $max_2 = 28$, **MAX** = 34;

$i_{min} = 2$, $min_1 = 32$, $min_2 = 25$, **MIN** = 25;

$cnt_L = 2$, $M = 2$, $M' = 3$, **MEDIAN** = 32.5;

$SUM(x^2) = 2 \times (2 + 30)^2 + 3 \times (3 + 30)^2 + 1 \times (4 + 30)^2 + (25^2 + 28^2) = 7880$; $E(x^2) = \frac{SUM(x^2)}{CNT}$; $E(x)^2 = MEAN^2$;

VAR = 8.4375; **STD** = 2.9;

$m1cnt = 3$; $i_{mode1} = 3$; $mode2 = 25$. (In *mSet*, 25 and 28 have the same frequency, and the first element, i.e., 25, is chosen as its mode.)

$m2cnt = 1$; because $m1cnt > m2cnt$, $\mathbf{MODE} = f_m^{-1}(i_{mode1}) = 33$.

Note that CNT is 8 instead of 10; this is because there are two nodes whose data is out of the operation range, which means there a failure is caused by node failure or other reasons. That is to say, computation of the final statistics, can automatically adapt to the dynamic network.

REDAR

In SEDAR, most elements of the encoded data $\vec{v}$ near the leaf nodes are zero, which means it contains redundant information. Directly transmitting these low information data using a full vector is too expensive. As these encrypted zeros are used to hidden the exact position of the encrypted non-zero elements. Encrypting all zeros is not necessary, especially when L large.

In this section, we propose REDAR, which can significantly reduce the communication cost with the trade-off of lower security on leaf node. In REDAR, all non-zero elements and a small number of random selected zero elements of the leaf nodes' vector are encrypted.

Random Encryption

Random selection zero-elements are used to reduce packet size, as well as provide security for the non-zero elements.

First, $\vec{v}$ are split the into several segments. Then, all non-zero elements and a small number of randomly chose zero elements are encrypted.

For example, in Fig 3-1, there are 27 elements in $\vec{v}$, each element contains 3 bits, each cipher element is encrypted from 2 elements, where the leftmost cipher element only contain 1 element in this case. $\vec{v}$ is split into 3 segments, each of the right two segments has 5 cipher elements at most, and the last segments has 4 cipher elements at most.

For each segment whose elements are zero, a random number r will generate between 1 and n_s, where n_s is the number of elements in the segment. Then the r rightmost elements of the segment are encrypted using the homomorphic encryption scheme. For example, in Fig 3-1, all elements of the leftmost and the rightmost segments are zero, i.e., $n_s = 4$ and $n_s = 5$ respectively, and thus one cipher element in the leftmost segment and two cipher elements in the rightmost segment are obtained.

For the segment containing a non-zero element, a random number r is generated, where r is between 0 and $n_s - p_y$, and p_y is the position of non-zero elements in the segment with respect to the right end. Then the $r + p_y$ rightmost elements of the segment are encrypted. For example, in Fig 3-1, the $2nd$ segment contains a non-zero element, and $p_y = 2$. Because $r = 0$ is return by the random function, only $p_y + r = 2$ leftmost elements are encrypted.

Packing and Unpacking

The packing step is used for constructing packet from the random encryption result. As show Fig 3-2, encrypted data in each segment are packed together in the original order, and a delimiter is added between adjacent segments. Because each cipher has the same size and only several leftmost elements are encrypted, the original encrypted vector can be reconstructed in the unpacking step.

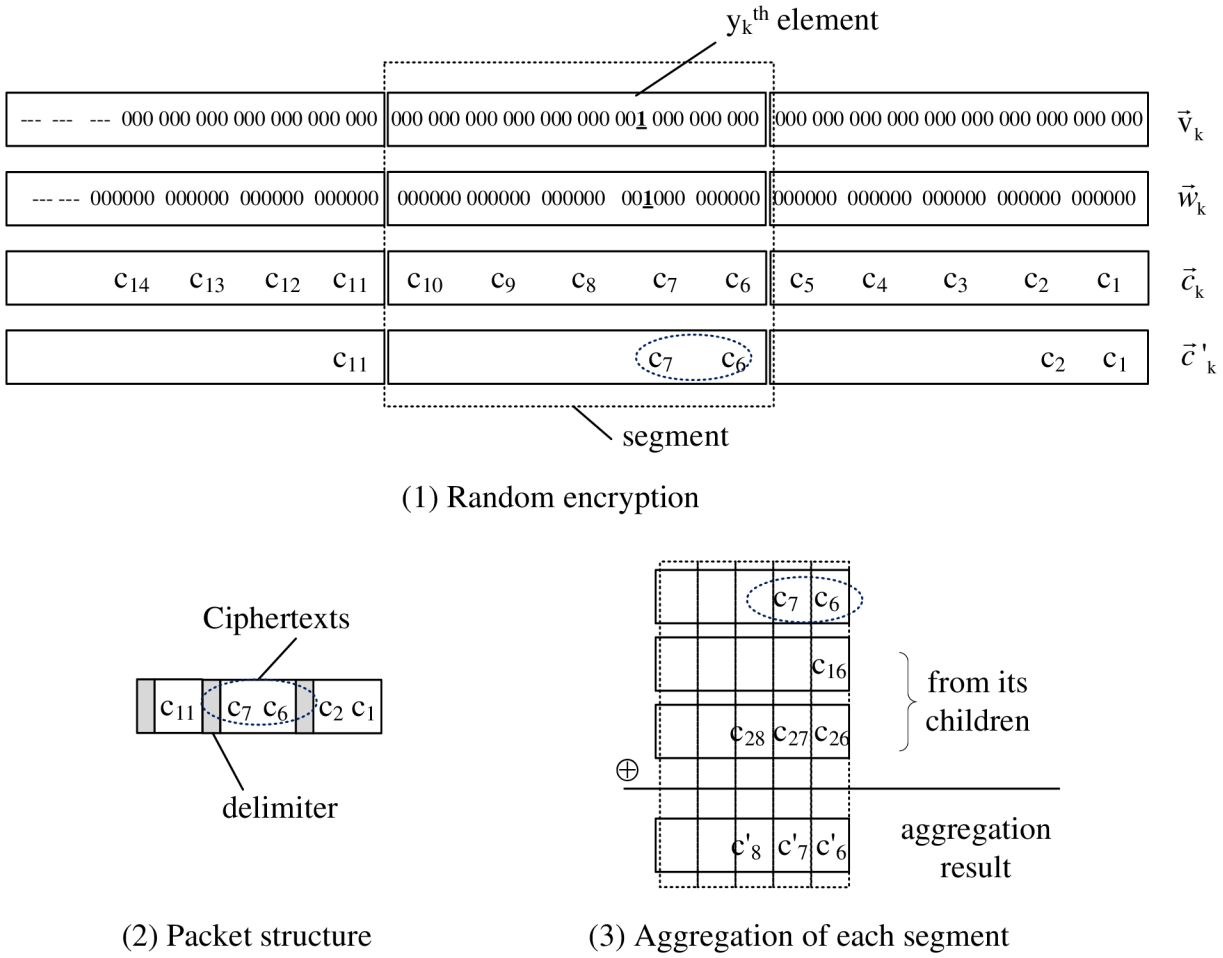

Fig 3. REDAR.

doi:10.1371/journal.pone.0159605.g003

Secure Data Aggregation

All received packets are unpacked to get $\vec{c}_k''$ sets, and then aligned together with the local generated encrypted data. Finally, data aggregation will performs directly on cipher domain column by column, and aggregation results of all segments are packed and send to its parent.

For example, in Fig 3-3, the first line is the encrypted data generated locally, the 2nd and 3rd are received from its children, and all of them are the 2nd segment of each vector. Segments received from different children may have a different number of elements, all of them are aligned to the right side. Because c_{17} is not exist in the 1st child, we ignore it, and just aggregate other two elements, i.e. $c_7' = c_7 \oplus c_{27}$. Among these three segments, the maximum number of elements is 3 (the 3nd line in Fig 3-3), so the elements number in the aggregation result of this segment is also 3.

Correctness and Security

As random selected encryption only performs on zero elements, all no-zero elements in each vector are encrypted, and aggregated into the final result. No raw data is loss in REDAR, so the final results are the same as that in SEDAR.

In REDAR, the communication cost is reduced for much less zero elements is encrypted and contained in the packet. However, as the number of encrypted data is reduced, it benefits the adversary with guessing the true position of non-zero elements. An inappropriate distribution of the encrypted data also benefits the adversary with the success probability of guess. So we need carefully design the random function and make sure that sufficient encrypted data is still retained after using random selected encryption. The more the encrypted elements, the lower probability it is guessed successfully. After aggregating at intermediate node, the encrypted elements will contain more than one encrypted non-zero elements, which means the success probability of the adversary will decrease significantly. More specifically, in the leaf node i, where only one non-zero elements in the vector. Assuming n_i encrypted data exist in the final packet, then the adversary's success probability is $\frac{1}{n_i}$. In the cluster header, assuming k nodes aggregate together, the probability reduces to $\frac{1}{n^k}$, where $n = \sum_j max_i(n_{ij})$, and n_{ij} is the number of elements in the jth segment of node i. As n and k increase along the aggregation tree, the adversary's success probability decrease obviously.

For example, when $n \geq 25$ and $k \geq 4$, the success probability no larger than 2.56×10^{-6}. As in cluster-based networks, the cluster member in each cluster is often large than 4, which means when $n \geq 25$, except in the leaf node, no encrypted data can be success guessed with probability larger than 2.56×10^{-6}. When k = 6 and n = 35, the success probability already decreases to 5.44×10^{-10}.

CEDAR

In this section, we present CEDAR. CEDAR and REDAR are complementary schemes. CEDAR is used before *encoding*, while REDAR is used after *encoding*.

In SEDAR, the total communication cost is mainly determined by the size and accuracy of R_d, i.e. $L = \frac{|R_d|}{a}$. L sometimes is large, so the total communication cost is still heavy. In order to reduce the communication cost, a *compression* step is introduced in CEDAR. As shown in Fig 4, mapping data y is compressed from a lager space with size of L into a smaller space with size of L'. Encoding step executes on compression data z, which make the vector length decreased from L to L'. Compression function can be a linear one or a non-linear one. Due to the limited space, we only illustrate the linear one.

A linear *compression* function f_c can compress y into z, i.e., $z = f_c(y) = f_c f_m(x) = \lceil \frac{y}{c} \rceil = \lceil \frac{f_m(x)}{c} \rceil$, c is the compression factor, which is larger than 1. Encoding step is based on z instead of y, i.e.,

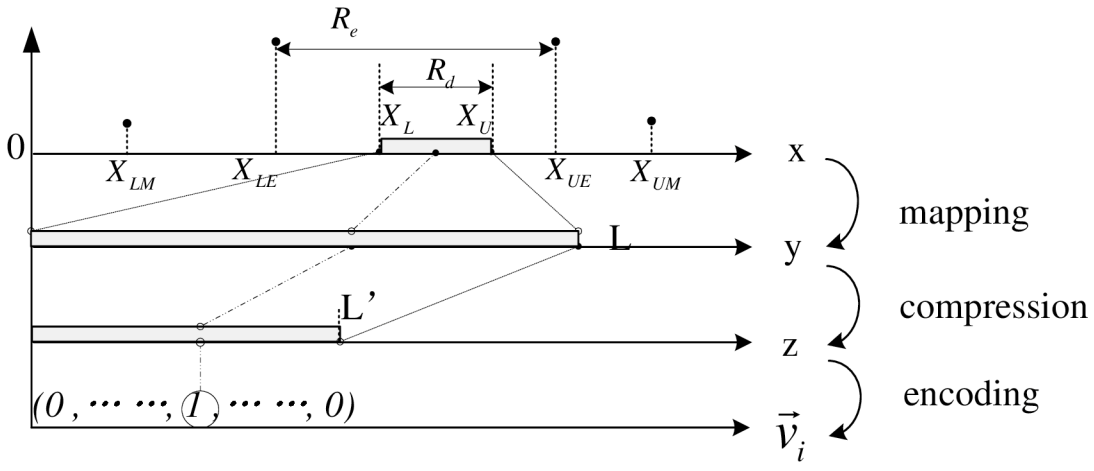

Fig 4. CEDAR.

doi:10.1371/journal.pone.0159605.g004

$\vec{v} = f_e(z) = f_e f_c(y)$. So the total communication cost is reduce from L to $\lceil \frac{L}{c} \rceil$. One can recover $\hat{y}$ as an estimate of y, using the decompressing function on z, i.e.,
$\hat{y} = f_c^{-1}(z) = f_c^{-1} f_e^{-1}(i) = c \times z - \lfloor \frac{\epsilon}{2} \rfloor = c \times i - \lfloor \frac{\epsilon}{2} \rfloor$.

Performance Analysis

In this section, we analyse communication and computation performance of proposed schemes. Performance criteria includes whether there are existing a bottleneck, and whether it achieves load balance.

Communication performance

First, we analyse the maximum packet size to judge whether exists a bottleneck. Then analyse the distribution attribution of packet size to judge whether it achieves load balance. After a thorough analysis, we find out that no bottleneck exist, and it achieves load balance. The detailed analysis goes as follows.

Data in R_d are encrypted by a homomorphic scheme, so encrypted data can aggregate directly in cipher domain in the intermediate nodes, and thus the total length will not change. Data in R_b uses a traditional encryption scheme. Without the private key, the intermediate nodes have to cascade each encrypted data and replay forwarding. So the length will increase, the minimum values of the data packets size appear in the leaf node of the aggregation tree, and the maximum length of the package is in the vicinity of the server node.

Now, let's analyse packet length for R_d and R_b respectively. For the sake of simplicity, data length analysis is based on plain text.

Communication cost for R_d is determined by the number of elements in the vector and the data length of each element. The former is determined by the range length of R_d and accuracy requirement a. The latter determined by the largest number of samples fall in the same point, and the worst case is all samples in R_d (i.e. $N \times P(R_d)$) fall in the same position. In practice, the probability of the worst case can be ignored. So,
$Cost_{R_d} < \frac{|R_d|}{a} \lceil log(N \times P(R_d)) \rceil = \frac{2\beta\delta}{a} \lceil log(N \cdot P(R_d)) \rceil$.

Now, let's consider the communication cost for R_b. Since the data in R_b is not aggregated in the intermediate nodes, the total data length will reach the maximum in the vicinity of the server node. The maximum value is determined by the total number of samples N and the probability ($P(R_b)$) of the sample in the region R_b, as well as the transmission overhead of a single sample. In practice, we can assume that the probability of abnormal data is much less than the normal one, which means the number of elements outside R_e can be ignored. So,
$P(R_b) = P(R_e \overline{R_d}) = P(R_e) - P(R_d) \approx 1 - P(R_d)$. and then
$Cost_{R_b} < \mathrm{N} \times \mathrm{P}(\mathrm{R}_b) \lceil \log \frac{R_b}{a} \rceil \approx \mathrm{N}(1 - P(\mathrm{R}_d)) \lceil \log \frac{|\mathrm{R}_b| - |\mathrm{R}_d|}{a} \rceil$.

In order to judge wether it achieves load balance, let's analyze the distribution of the whole network traffic first.

The minimum values of the data packets size appear in the leaf node of the aggregation tree. In the leaf node, when $x \in R_b$, no encoding step is used, so its data length is $\lceil log \frac{|R_e| - |R_d|}{a} \rceil$. When $x \in R_d$, the encoding data length is $Cost_{R_d}$. The former one (Let's denote it as C_0) is much less than the last one. However, C_0 only exists in very small number of leaf node. Assuming each cluster contains m leafs. The probability that C_0 appears at the same time in m nodes is $(1 - P(R_d))^m$. For example, in a normal distribution, assuming $\beta = 2$, when m = 3, $(1 - P(R_d))^m = 9.48 \times 10^{-5}$. The probability is small, which means even a small number of leaf node has a packet size of C_0, its parent will at least $Cost_{R_d}$. Therefore, C_0 is lack of significance, and the representative minimum value should be selected as $Cost_{min} = Cost_{R_d}$.

In the whole network, the minimum packet appears in the leaf nodes, the maximum packet appears in the vicinity of the server. In the path of the leaf node to the root node, the size of the packet increases from the minimum to the maximum value. Because $P(R_d) \gg P(R_b)$, the growth rate of packet size is small enough, and the average packet size $\overline{Cost} \approx Cost_{R_d}$.

The maximum value of the data packet is, $Cost_{\max} = Cost_{R_d} + Cost_{R_d}$. On the one hand, once the R_d is determined, $Cost_{R_d}$ can be treat as a const. If $Cost_{R_d}$ is too large, CEDAR can be used. So $Cost_{R_d}$ is controllable. On the other hand, according the define of R_d, $P(R_d) \approx 1$, so $Cost_{R_d}$ is small enough. As a result, the maximum packet size can be regarded as a controllable const, and there is no bottleneck in the network.

As the difference among $Cost_{min}$, $Cost_{max}$ and $\overline{Cost}$ is small, we can easily make a conclusion that it achieves load balance.

Computation performance

For computation performance, we also analyse the maximum computation cost to judge whether exists a bottleneck, and analyse the distribution attribution of computation cost to judge whether it achieves load balance. We find out that no bottleneck exist, and it achieves load balance. The detailed analysis goes as follows.

Each data can either be homomorphic encrypted after encoded, or encrypted by traditional scheme directly, according the range it belongs to. Let's denote the computation cost of the former as C_{11}, and the latter one as C_{21}.

For the data inside the R_d, mapping and encoding are required before homomorphic encryption. Both of them cost much less than encryption and decryption, thus can be ignored. For the homomorphic encrypted data, the intermediate nodes will not decrypt it, and aggregate them directly in cipher domain. Assuming a single cipher domain addition cost C_{13}, the total aggregation cost is $C_{13}(N \times P(R_d) - 1)$, due to that $N \times P(R_d) - 1$ times aggregation operation are necessary for $N \times P(R_d)$ elements in R_d.

The final aggregated result will be decrypted in the server, and the decryption cost is C_{12}. Each encrypted data in the boundary range, will also decrypted in the server, and the decryption cost is C_{22}.

So the total computational cost is $C = C_{11}\, N \times P(R_d) + C_{13}(N \times P(R_d) - 1) + C_{12} + N \times P(R_b)$ $(C_{21} + C_{22})$.

Average computational cost is $C_{avg} = \frac{C}{N} = C_{11}P(R_d) + \frac{C_{13}(N \times P(R_d) - 1)}{N} + \frac{C_{12}}{N} +$ $P(R_b)\,(C_{21} + C_{22}) \approx (C_{11} + C_{13})\,P(R_d) + (C_{21} + C_{22})\,(1 - P(R_d))$.

In instances of homomorphic encryption, encryption cost and decryption cost are often much larger than the cipher domain aggregation cost. For example, in the ECC-based version, the main operations of encryption and decryption are scalar multiplication, and the main operation of cipher domain aggregation is point addition. The former is far greater than the latter, so C_{13} can also be ignored, and $C_{avg} \approx C_{11}\, P(R_d) + (C_{21} + C_{22})(1 - P(R_d))$.

There are two main computational cost operations, i.e. encryption and decryption. Both of them not exist in the same node. Each client only performs one type of encryption operation, i.e. homomorphic one or traditional one. Two types of decryption exist in the server.

Since each client only chooses one of the two kinds of encryption mechanisms, each data is encrypted only once, so the computation cost is C_{11} or C_{21}. In general, the encode data has larger length than the raw data, so $C_{11} > C_{21}$, so the maximum computational cost of client is C_{11}. Two types of decryption exist in the server, the corresponding overhead is $C_{12} + N \times P(R_b)C_{22}$. Because $N \times P(R_b)$ is often small, and the server node has a large computational power, the decryption operation is not a difficult task. Therefore, there is no computational bottleneck exist.

Table 2. Comparison on Statistics Functions and Encoding Method.

	Encoding	Statistics
Considine et al. [9]	Synopsis generation function	CNT, SUM, AVG VAR, STD
Roy et al. [11]		
Li et al. [17]	Slicing and assembling technique	
Yang et al. [18]		
Castelluccia et al. [10]	No	
Lu et al. [19]		
Ertaul et al. [20]	No	MAX, MIN
Samanthula et al. [21]		
RCDA [14]	$l = \lceil logL \rceil$, $\beta = l(i-1)$, $\vec{v}_i = x_i \| 0^{\beta}$	CNT, SUM, AVG VAR, STD, MAX MIN, MODE, MEDIAN
EERCDA [13]		
Proposed schemes	$\vec{v}_i = zeros(1, L), \vec{v}_i(y_i) = 1$	

doi:10.1371/journal.pone.0159605.t002

Each client only encryption once, and the aggregation operations in each intermediate node are not compute-intensive, so the the proposed scheme also achieve load balance in computation.

Statistics functions supported

In this section, we compare the proposed schemes with other data aggregation schemes on statistics functions and encoding method. Table 2 is the comparison result. All of them are distributed aggregation schemes, which mean that intermediate nodes generate partial aggregation results from their received data.

Evaluation

Data sets description

Evaluation is based on six datasets gathered from different type of sensors. All of them are obtained from TAO (Tropical Atmosphere Ocean) project. The TAO is a project of NOAA (National Oceanic and Atmospheric Administration), which aim to enable real-time collection of high quality oceanographic and surface meteorological data for monitoring, forecasting, and understanding of climate swings associated with El Nino and La Nina.

Table 3 is the general description of each dataset. *Rh0n156e_hr* is a dataset of relative humidity. *Bp0n156e_hr* is sea level pressure. *W0n156e_hr* is wind direction. *Sst0n147e_hr* and *sst0n156e_hr* are different datasets of sea surface temperature. *rad0n156e_hr* is shortwave radiation. The 2nd column is the sample size of each dataset. The 3rd and 4th columns are skewness and kurtosis respectively. The 5th and 6th columns are mean and standard deviation estimated using the history record.

Table 3. General description of datasets.

Dataset	Size	Skewness	Kurtosis	$\hat{\delta}$	$\hat{\mu}$
rh0n156e_hr	144250	0.331	3.301	5.508	78.587
bp0n156e_hr	122035	-0.174	2.852	1.820	1008.3
w0n156e_hr	146702	-0.409	1.992	96.268	197.416
sst0n147e_hr	65535	0.313	3.706	0.469	29.712
rad0n156e_hr	39916	-0.008	1.757	234.273	596.817
sst0n156e_hr	4672	0.829	4.683	0.358	29.338

doi:10.1371/journal.pone.0159605.t003

Table 4. $P(R_b)$ in different dominant range setting.

Dataset	$\beta = 1.4$	$\beta = 1.8$	$\beta = 2.2$	$\beta = 2.6$	$\beta = 3$
rh0n156e_hr	15.61%	7.39%	3.22%	1.40%	0.50%
bp0n156e_hr	16.40%	6.78%	2.54%	0.66%	0.19%
w0n156e_hr	17.11%	4.82%	0	0	0
sst0n147e_hr	14.10%	6.73%	3.27%	1.55%	0.75%
rad0n156e_hr	18.70%	0	0	0	0
Sst0n156e_hr	14.75%	5.78%	3.47%	1.85%	1.02%

doi:10.1371/journal.pone.0159605.t004

These datasets cover several different scenarios. The distribution characteristics of them are different from each other, and thus has certain representativeness.

Effectiveness of Range Segmentation Model

In the proposed schemes, a range segmentation model is introduced to reduce the encoded vectors length, and thus reduce the total communication cost as long as the $P(R_b)$ is small enough. To achieve this purpose, we should choose dominate range carefully and make sure that samples beside this range is small enough. Now, let's verify whether the boundary setting of the dominate range (R_d) is effective.

As we described above, the lower bound X_L and the upper bound X_U of dominate range R_d are determined by $X_L = max(X_{LE}, \hat{\mu} - \beta\hat{\delta})$ and $X_U = min(X_{UE}, \hat{\mu} + \beta\hat{\delta})$. X_{LE} and X_{UE} are const defined in TAO project. $\hat{\mu}$ and $\hat{\delta}$ are estimated from history data, which can also be regarded as const. Different dominate range can be generated by different β.

The proportion of data outside R_d, i.e. $P(R_b)$, under different parameters are list in Table 4. As β increase, $P(R_b)$ reduce significantly. For example, when $\beta = 2.2$, $P(R_b)$ is no larger than 3.5% in all six datasets, and two of them are even reduced to zero.

However, it's not means the larger of β, the better of the communication performance. As β increase, the encoded vector length also increase, and the communication cost for R_d will increase. We need to achieve a balance between the communication cost for R_d and R_b.

Fig 5 is the relationship between the maximum packet size and dominant range R_d setting. The network size is 1000. We can easily find out that, as β increasing, the maximum packet size reduces significantly, and when $\beta > 1.8$, the decreasing rate becomes moderate. Let's analysis the reason. As the increase of β, the communication cost for the boundary range R_b reduce significantly. More specifically, the communication overhead reduced in R_b is much larger than that increased in R_d, so the total packet size is still significantly decreased. When β reaches a certain value, the maximum packet size reaches a minimum value. E.g., $\beta = 1.6$ for *sst0n156e_hr* and $\beta = 2$ for *w0n156e*. In some case, when β is larger than its optimal value, $P(R_b)$ is small enough, the communication cost for R_b can be ignored, and the cost for R_d is may increase as the bound of R_d still in R_e, so the maximum packet size may increase mildly.

Communication Cost of SEDAR in Different R_d Setting

Fig 5 is the communication cost of SEDAR in different R_d setting. According to this figure, when β is between 1.8 and 3, the change of the maximum packet size in each dataset is relatively small. Which means, any $\beta \in [1.8, 3]$ meets the basically requirements and doesn't significantly reduce the communication performance. This feature is very useful, which means R_d setting is easy.

Although it is difficult to achieve optimal performance by setting an accurate dominant range in advance, by choosing arbitrary $\beta \in [1.8, 3]$, we can still obtain a suboptimal

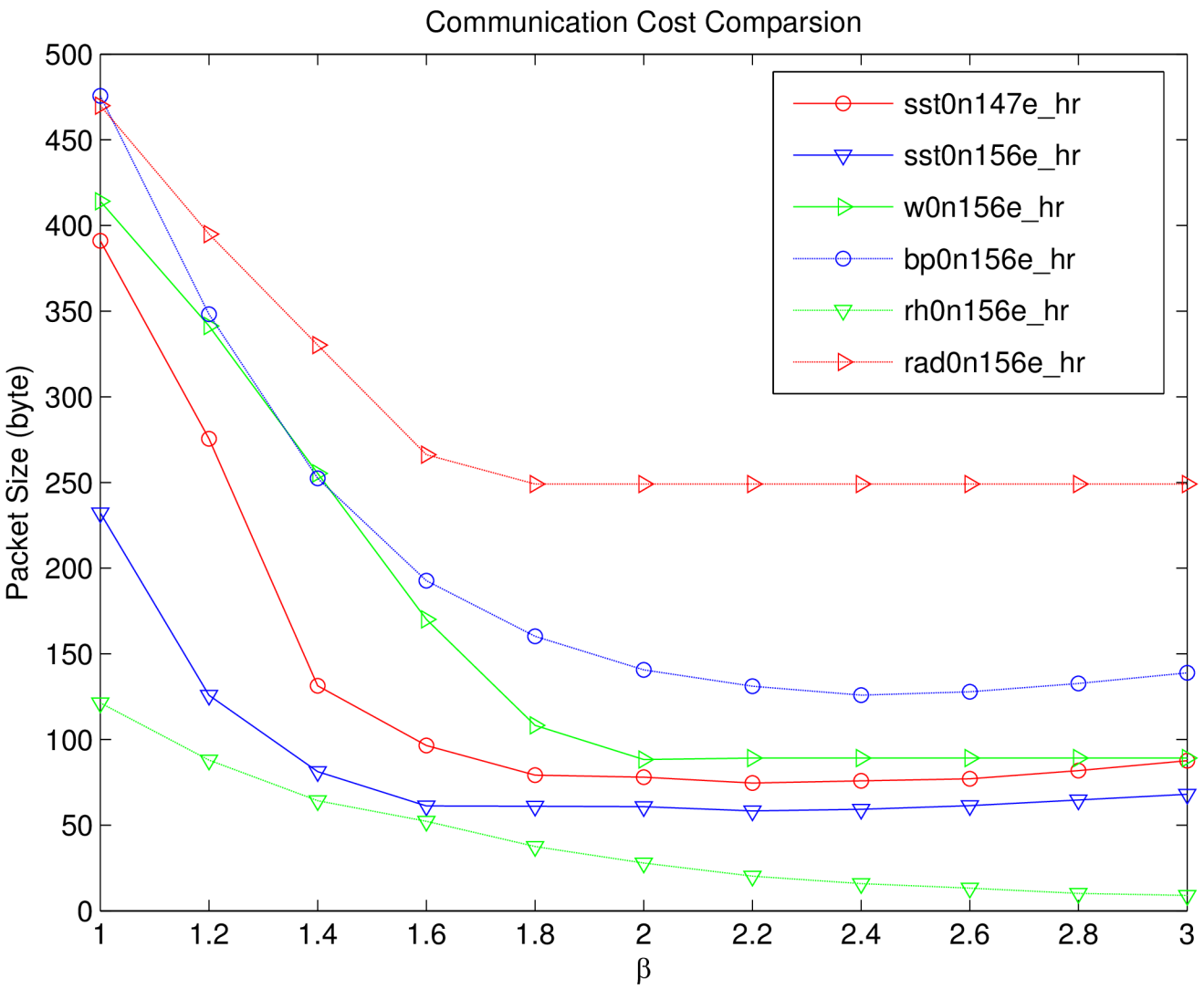

Fig 5. Communication cost in different R_d setting.

doi:10.1371/journal.pone.0159605.g005

performance, which is very similar to the optimal one. For example, in following evaluation, we directly set $\beta = 2$ for different datasets, and still obtain a good result.

Comparsion with RCDA and EERCDA

In this section, we compare SEDAR with RCDA and EERCDA. Both of them support mutil-functional security data aggregation, and homomorphic encryption scheme is used in all of them. The main difference lies in the encoding function. Each client encrypts collected and encoded data. The aggregation performs on cipher text directly at each intermediate aggregator, and the decryption performs on the server. The computation cost of encryption and decryption are near-linear related to the encoded data length. Limited to space, we only concern the comparison on communication cost. In these evaluations, $\beta = 2$. The comparison result lists in Fig 6. (θ is used to characterize the intensity of data fluctuation in a given application for EERCDA.)

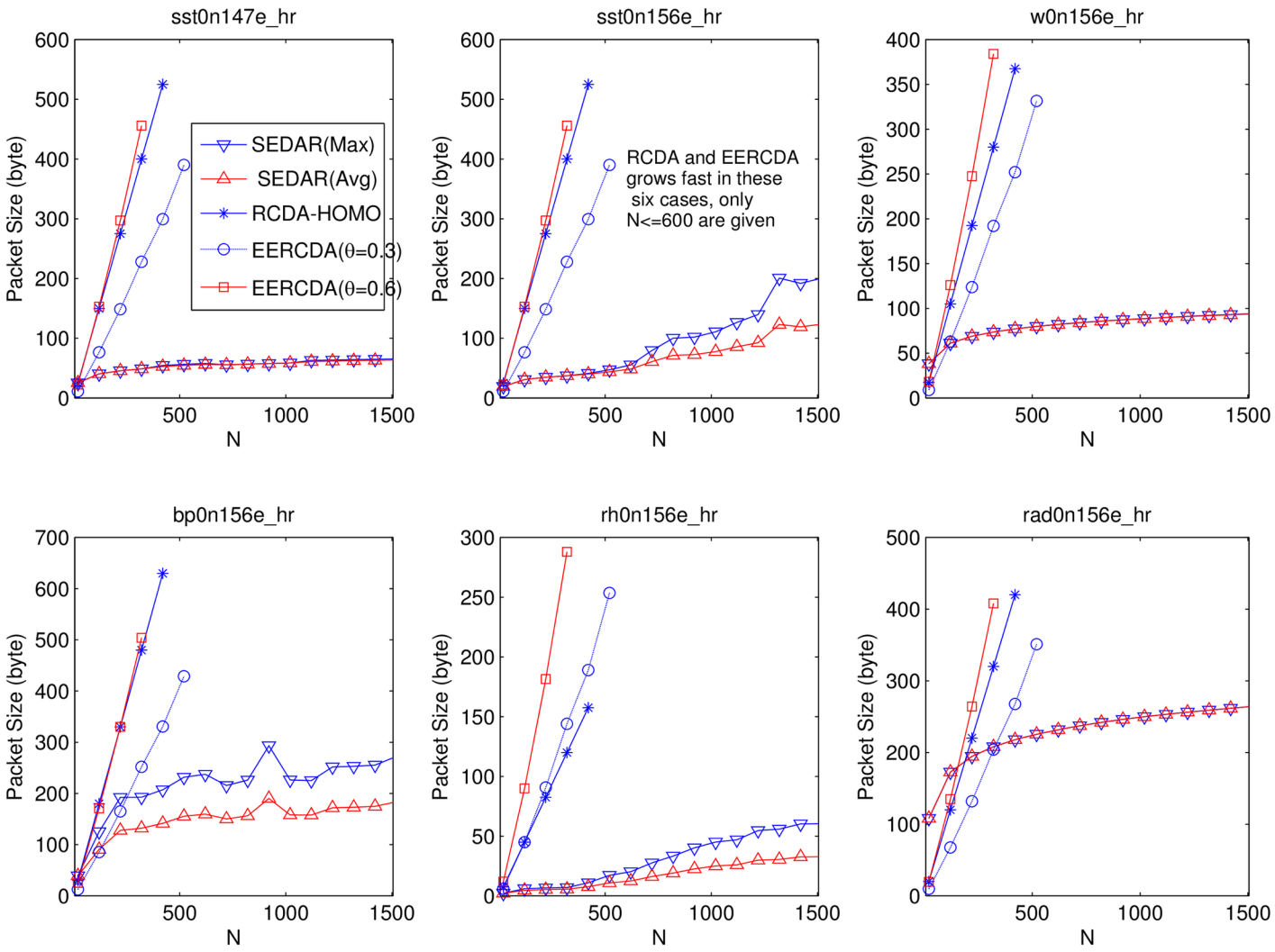

Fig 6. Comparsion with RCDA and EERCDA.

doi:10.1371/journal.pone.0159605.g006

According Fig 6, we can easily find out that the proposed scheme is obviously superior to RCDA and EERCDA. And due to the slow growth of communication cost as the increase of N, it can be applied to large scale networks.

In *w0n156e_hr*, *bp0n156e_hr* and *rad0n156e_hr*, when N is small, RCDA and EERCDA is better than SEDAR. In these datasets, when $\beta = 2$, the dominate range is a little large. So when the network size N is small, the communication cost is larger than that in RCDA and EERCDA. However, when N increases to a certain value, the advantage of this scheme is very obvious. In other three datasets, the dominate region size is small, and most samples are in the dominate range when $\beta = 2$, so the proposed scheme has an absolute advantage even in the small network.

In *sst0n147e_hr*, *w0n156e_hr* and *rad0n156e_hr*, the average and maximum communication cost are almost have the same value, this is because most elements are in R_d. In contrast, the average and maximum communication cost aren't the same in other three datasets.

Table 5 is comparison on end-to-end aggregation time. Due to limited space, we only compare SEDAR with RCDA. The evaluation is built on MICAz. MICAz has a low-power 8-bit microcontroller ATmega128L and an IEEE 802.15.4 compliant CC2420 transceiver. The clock frequency of

Table 5. Comparison on end-to-end aggregation time (unit: ms; N = 1000; cluster-based network. Compu.: computation delay; Commu.: communication delay; Total: total delay).

	SEDAR			RCDA		
	Compu.	Commu.	Total	Compu.	Commu.	Total
sst0n147e_hr	1.52×10^4	15.59	1.52×10^4	4.96×10^5	339.25	4.96×10^5
sst0n156e_hr	1.15×10^4	20.63	1.15×10^4	4.35×10^5	339.25	4.35×10^5
w0n156e_hr	2.30×10^4	23.54	2.30×10^4	4.37×10^5	237.48	4.37×10^5
bp0n156e_hr	2.33×10^4	42.06	2.34×10^4	7.56×10^5	407.11	7.56×10^5
rh0n156e_hr	1.29×10^3	6.65	1.29×10^3	7.97×10^4	101.77	7.98×10^4
rad0n156e_hr	6.50×10^4	66.48	6.51×10^4	1.05×10^6	271.40	1.05×10^6

doi:10.1371/journal.pone.0159605.t005

ATmega128L is 8 MHz. The claimed data rate of CC2420 is 250 kbps. Meulenaer et al. [22] measured the effective data rate for transmitting is 121 kbps which far below the claimed rates. In the energy models used in this paper, we use 121 kbps as the data rate for the evaluation of communication delay. For the computational cost evaluation, we decide to implement the proposed scheme based on TinyECC [23]. According to its evaluation result based on MICAz, the execution time for encryption is 3907.46ms, which is similar to the one used in RCDA [14]. According to RCDA, MICAz needs 73.71 ms to aggregate two data in cipher domain. According to the comparison results, end-to-end aggregation time of SEDAR is much smaller than that of RCDA. With the increase of network size, this advantage will be more obvious.

Cost and Accuracy Evaluation for CEDAR

We measure the performance of CEDAR in this section. Figs 7 and 8 are the communication overhead and accuracy in different compression factor c.

According to Fig 7, we can know that with the increase of c, the average and the maximum communication cost of each dataset are reduced to some extent. The reduction trend in each dataset is not entirely consistent. When the compression factor is large, the communication volume curve is flat, which means the compression effect is decreased. According to Fig 8, with the increase of C, the error rates in each dataset are increasing. So it is necessary to ensure that a balanced between the communication cost and the error rate.

The growth trend of the error rate has a certain degree of relationship with the initial communication cost. The initial communication cost is the corresponding communication cost in SEDAR. The error rates increase more quickly in cases that have much larger initial communication cost. For example, as shown in Fig 8, the initial communication cost of *rad*0*n*156*e_hr* is relatively large, and when $c = 5$, the bound of error rates is still less than ±0.015. In *sst*0*n*147*e_hr*, *w*0*n*156*e_hr*, and *rh*0*n*156*e_hr*, in which the initial communication cost is small, the bound is close to or more than *pm*0.03 when $c = 5$. In particular, error bound of *rh*0*n*156*e_hr*, is greater than 0.4 when $c = 4$. In fact, according to Fig 7, the initial communication cost of *rh*0*n*156*e_hr* is the minimal one, and the reduction tendency of *rh*0*n*156*e_hr* is not obvious.

Hence, we can choose a large compression factor for the case with large initial communication cost, and we should choose a small one, or even give up the CEDAR for the case with small initial communication cost.

Related work

Distributed Aggregation

Distributed aggregation is a traditional research topic in database community. Kuhn and Oshman [6] studied the complexity of computing count and minimum in synchronous directed

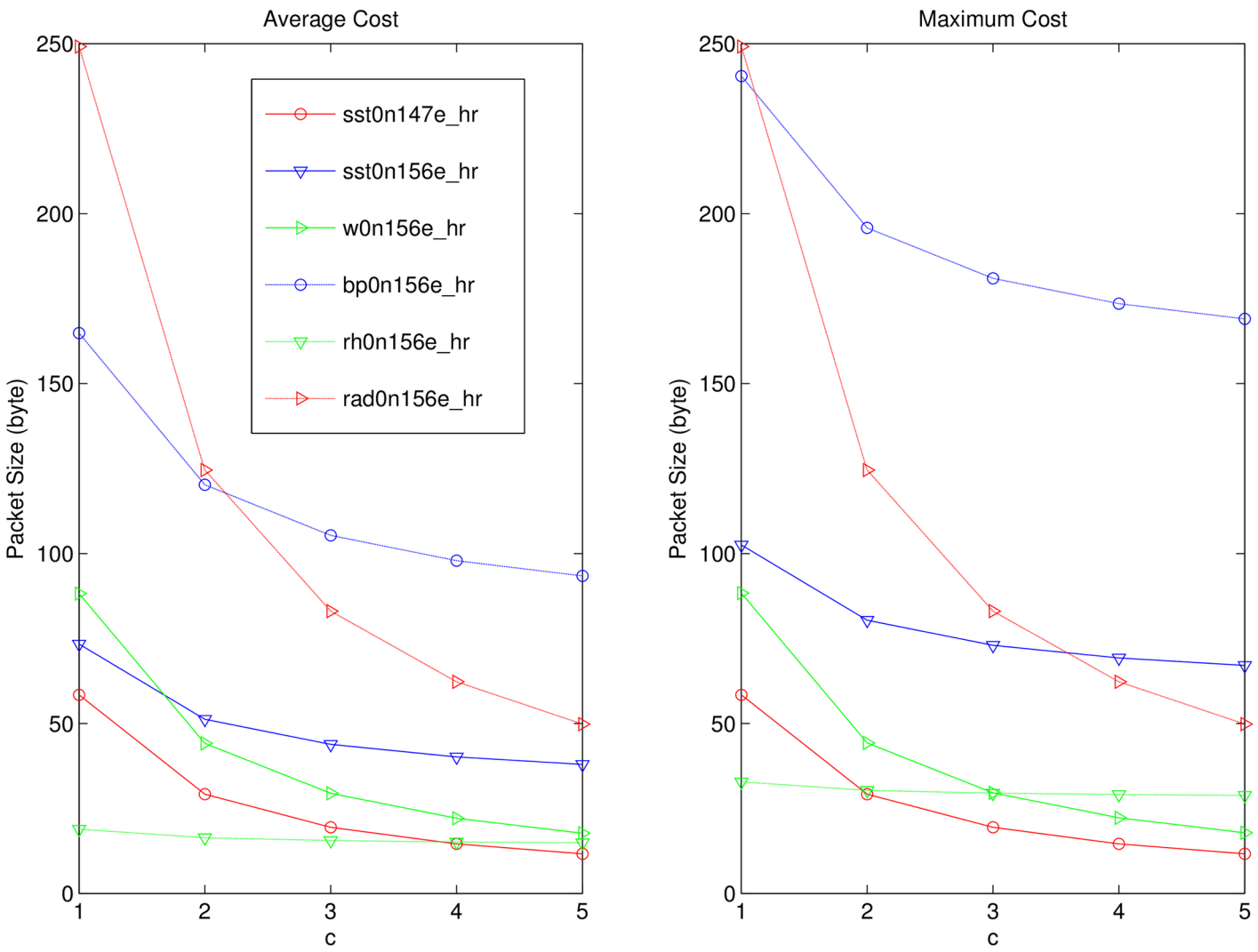

Fig 7. Cost Evaluation of CEDAR.

doi:10.1371/journal.pone.0159605.g007

networks. Hobbs et al. [7] presented a distributed protocol to compute maximum and average under the SINR model. Cormode and Yi [8] focused on tracking the value of a aggregation function on distributed monitoring area. Cheng et al. [24], Li and Cheng [25] considered the approximate aggregation problem and presented (ϵ, δ)-approximate schemes based on Bernoulli sampling. Xie and Wang [26] and Shen et al. [27] studied network construction and message routing algorithm for data aggregation.

Secure Distributed Aggregation

Several secure distributed aggregation schemes have been proposed. Most of them focus on secure itself, and very limited numbers of aggregation functions can be supported. Considine et al. [9] and Roy et al [11] proposed secure distributed aggregation scheme for duplicate sensitive aggregation based on synopsis generation function. Li et al. [17] and Yang et al. [18] proposed slice-mix based schemes for additive aggregation functions, which guarantees data privacy through data "slicing and assembling" technique. Castelluccia et al. [10] and Lu et al.

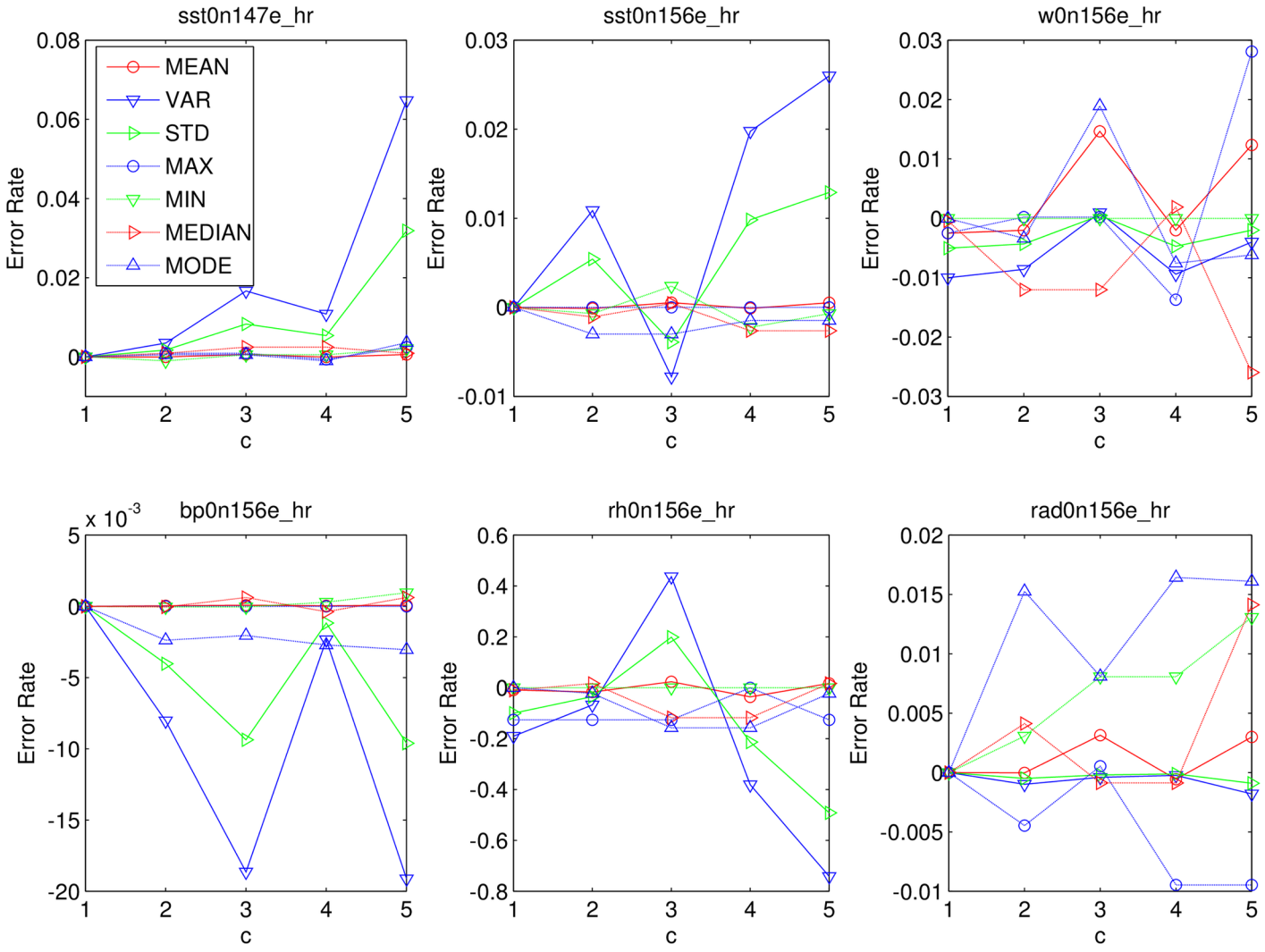

Fig 8. Accuracy Evaluation of CEDAR.

doi:10.1371/journal.pone.0159605.g008

[19] proposed secure distributed aggregation scheme based on homomorphic encryption, which is also only support summation-based statistical functions, such as CNT and SUM. Agrawal et al. [28] presented the order-preserving encryption scheme. Ertaul et al. [20] and Samanthula et al. [21] applied it to secure distributed aggregation, to get comparison-based statistics, such as MAX, MIN. However, summation-based statistics is not support in these schemes. Chien-Ming et al. [14] and Jose et al. [13] adopted encoding steps before encryption to achieve arbitrary aggregation function. However, their encoding steps are simple concatenation all sensing data without any information compression method, and the communication cost is too heavy to extend to large scale networks. Enabling operation in cipher domain is also an important topic in cloud computing [29–31]. In addition to traditional encryption scheme, data privacy can be achieved by steganography [32, 33]. Beside data privacy, date authentication is also necessary. Ren et al. [34] proposed an efficient mutual verifiable provable data possession scheme. Guo et al. [35] designed a lightweight and tolerant authentication to guarantee data security.

Conclusions

In this paper, we have studied the problem of *multifunction secure distributed aggregation*, and also have proposed three complementary schemes (i.e., SEDAR, REDAR and CEDAR) to solve this problem. The first one can obtain accurate aggregation results. The other two can significantly reduce communication cost with the trade-off lower security and lower accuracy, respectively. Extensive analysis and experiments, based on six different scenes of real data, have shown that all of them have an excellent performance.

Acknowledgments

This work is supported by the Foundation of Hunan Educational Committee (14C0484), National Natural Science Foundation of China under Grant (61502054), Yongzhou Science and Technology Plan ([2013]3), Open Research Fund of Hunan Provincial Key Laboratory of Network Investigational Technology (2016WLZC016), Foundation of Hunan University of Science and Engineering (13XKYTA003). The authors declare that they have no conflict of interests.

Author Contributions

Conceived and designed the experiments: PZ.

Performed the experiments: PZ.

Analyzed the data: PZ WJL HS.

Contributed reagents/materials/analysis tools: PZ WJL HS.

Wrote the paper: PZ WJL HS.

References

1. Yan Y, Zhang J, Huang B, Sun X, Mu J, Zhang Z, et al. Distributed Outlier Detection using Compressive Sensing. In: SIGMOD'15. ACM; 2015. p. 3–16.
2. Shim KA, Park CM. A Secure Data Aggregation Scheme based on Appropriate Cryptographic Primitives in Heterogeneous Wireless Sensor Networks. IEEE Transactions on Parallel and Distributed Systems. 2014; PP(99):1–1.
3. Han J, Kamber M, Pei J. Data Mining: Concepts and Techniques, 3rd ed. Morgan Kaufmann; 2011.
4. Wuhib F, Stadler R, Spreitzer M. A Gossip Protocol for Dynamic Resource Management in Large Cloud Environments. IEEE Transactions on Network and Service Management. 2012; 9(2):213–225. doi: 10.1109/TNSM.2012.031512.110176
5. Zhang G, Liu W, Hei X, Cheng W. Unreeling Xunlei Kankan: Understanding Hybrid CDN-P2P Video-on-Demand Streaming. IEEE Transactions on Multimedia. 2015; 17(2):229–242. doi: 10.1109/TMM.2014.2383617
6. Kuhn F, Oshman R. The complexity of data aggregation in directed networks. In: DISC'11. Springer; 2011. p. 416–431.
7. Hobbs N, Wang Y, Hua QS, Yu D, Lau FC. Deterministic distributed data aggregation under the SINR model. In: TAMC'12. Springer; 2012. p. 385–399.
8. Cormode G, Yi K. Tracking distributed aggregates over time-based sliding windows. In: SSDBM'12. Springer; 2012. p. 416–430.
9. Considine J, Hadjieleftheriou M, Li F, Byers J, Kollios G. Robust approximate aggregation in sensor data management systems. ACM Transactions on Database Systems. 2009; 34(1):6. doi: 10.1145/1508857.1508863
10. Castelluccia C, Chan ACF, Mykletun E, Tsudik G. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. ACM Transactions on Sensor Networks. 2009; 5(3):1–36. doi: 10.1145/1525856.1525858

11. Roy S, Conti M, Setia S, Jajodia S. Secure Data Aggregation in Wireless Sensor Networks. IEEE Transactions on Information Forensics and Security. 2012; 7(3):1040–1052. doi: 10.1109/TIFS.2012.2189568
12. Lin YH, Chang SY, Sun HM. CDAMA: Concealed Data Aggregation Scheme for Multiple Applications in Wireless Sensor Networks. IEEE Transactions on Knowledge and Data Engineering. 2013; 25 (7):1471–1483. doi: 10.1109/TKDE.2012.94
13. Jose J, Manoj Kumar S, Jose J. Energy efficient recoverable concealed data aggregation in wireless sensor networks. In: ICE-CCN'13. IEEE; 2013. p. 322–329.
14. Chien-Ming C, Yue-Hsun L, Ya-Ching L, Hung-Min S. RCDA: Recoverable Concealed Data Aggregation for Data Integrity in Wireless Sensor Networks. IEEE Transactions on Parallel and Distributed Systems. 2012; 23(4):727–734. doi: 10.1109/TPDS.2011.219
15. Fousse L, Lafourcade P, Alnuaimi M. Benaloh's dense probabilistic encryption revisited. In: Progress in Cryptology–AFRICACRYPT 2011. Springer; 2011. p. 348–362.
16. Guellier A. Can Homomorphic Cryptography ensure Privacy? [Research Report]. IRISA; Supélec Rennes, équipe Cidre; Array; 2014.
17. Li H, Lin K, Li K. Energy-efficient and high-accuracy secure data aggregation in wireless sensor networks. Computer Communications. 2011; 34(4):591–597. doi: 10.1016/j.comcom.2010.02.026
18. Yang G, Li S, Xu X, Dai H, Yang Z. Precision-enhanced and encryption-mixed privacy-preserving data aggregation in wireless sensor networks. International Journal of Distributed Sensor Networks. 2013;. doi: 10.1155/2013/427275
19. Lu M, Shi Z, Lu R, Sun R, Shen XS. PPPA: A practical privacy-preserving aggregation scheme for smart grid communications. In: (ICCC'13. IEEE; 2013. p. 692–697.
20. Ertaul L, Kedlaya V. Computing Aggregation Function Minimum/Maximum using Homomorphic Encryption Schemes in Wireless Sensor Networks. In: ICWN'07. IEEE; 2007. p. 186–192.
21. Samanthula BK, Jiang W, Madria S. A Probabilistic Encryption Based MIN/MAX Computation in Wireless Sensor Networks. In: MDM'13. IEEE; 2013. p. 77–86.
22. De Meulenaer G, Gosset FCCO, Standaert FCCOX, Pereira O. On the energy cost of communication and cryptography in wireless sensor networks. In: WiMob 2008;.
23. Liu A, Ning P. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In: IPSN'08. IEEE; 2008. p. 245–256.
24. Cheng S, Li J, Ren Q, Yu L. Bernoulli Sampling Based (ϵ, δ)-Approximate Aggregation in Large-Scale Sensor Networks. In: INFOCOM'10. IEEE; 2010. p. 1–9.
25. Li J, Cheng S. (ϵ, δ)-Approximate Aggregation Algorithms in Dynamic Sensor Networks. IEEE Transactions on Parallel and Distributed Systems. 2012; 23(3):385–396. doi: 10.1109/TPDS.2011.193
26. Xie S, Wang Y. Construction of tree network with limited delivery latency in homogeneous wireless sensor networks. Wireless personal communications. 2014; 78(1):231–246. doi: 10.1007/s11277-014-1748-5
27. Shen J, Tan H, Wang J, Wang J, Lee S. A novel routing protocol providing good transmission reliability in underwater sensor networks. Journal of Internet Technology. 2015; 16(1):171–178.
28. Agrawal R, Kiernan J, Srikant R, Xu Y. Order preserving encryption for numeric data. In: SIGMOD'04. ACM; 2004. p. 563–574.
29. Fu Z, Ren K, Shu J, Sun X, Huang F. Enabling Personalized Search over Encrypted Outsourced Data with Efficiency Improvement. IEEE Transactions on Parallel and Distributed Systems. 2015; PP(99):1–1.
30. Xia Z, Wang X, Sun X, Wang Q. A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data. IEEE Transactions on Parallel and Distributed Systems. 2016;. doi: 10.1109/TPDS.2015.2401003
31. Fu Z, Sun X, Liu Q, Zhou L, Shu J. Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. IEICE Transactions on Communications. 2015; 98(1):190–200. doi: 10.1587/transcom.E98.B.190
32. Xia Z, Wang X, Sun X, Wang B. Steganalysis of least significant bit matching using multi-order differences. Security and Communication Networks. 2014; 7(8):1283–1291. doi: 10.1002/sec.864
33. Xia Z, Wang X, Sun X, Liu Q, Xiong N. Steganalysis of LSB matching using differences between nonadjacent pixels. Multimedia Tools and Applications. 2016; 75(4):1947–1962. doi: 10.1007/s11042-014-2381-8
34. Ren Y, Shen J, Wang J, Han J, Lee S. Mutual verifiable provable data auditing in public cloud storage. Journal of Internet Technology. 2015; 16(2):317–323.
35. Guo P, Wang J, Li B, Lee S. A variable threshold-value authentication architecture for wireless mesh networks. Journal of Internet Technology. 2014; 15(6):929–936.