Secure Data Aggregation using Multiparty Computation

Vimal Kumar  
Computer Science Department  
vkq3f@mst.edu

Sanjay Madria  
Computer Science Department  
madrias@mst.edu

Abstract

Wireless sensors are usually deployed in hostile environments where they are easily accessible to the adversary. Security of the sensors and integrity of the data becomes an important issue in this scenario. In this paper we propose a scheme for data aggregation, using secure multiparty computation concepts. Our scheme, ensures the correctness of the data received by the base station and also ensures the privacy of the individual data of sensor nodes.

1. Introduction

A wireless sensor network (WSN) is a spatially distributed and self organizing network of a large number of low cost sensors. WSNs have emerged as a popular solution to many applications in both controlled and uncontrolled environments in fields as varied as environmental sensing, wildlife monitoring, building safety, traffic monitoring, law enforcement and military among others. The sensors used in these networks have resource constraints which make WSNs different from traditional networks.

1. Sensors are typically battery operated devices, which means energy is limited.
2. Sensors usually have limited RAM, and restricted bandwidth available and broadcast is the means of communication among nodes.
3. Sensors are usually deployed at readily accessible places, which means they are not physically secure.

These constraints imply that protocols used in traditional networks cannot be blindly used in sensor networks.

In order to reduce energy consumption of sensors, data aggregation is used. In data aggregation a small amount of computation is done in the network to save substantial amount of energy. Data aggregation provides us with savings in communication but on the other hand throws at us new challenges in data aggregation security. Aggregator security becomes an important issue in these types of networks. An aggregator which is compromised by an adversary may continue to work in the same manner as before, but can inject false data to the aggregated value. This is known as false data injection attack, and can be difficult to identify in data aggregation setup. To counter this attack, we need a trusted setup among the sensor nodes, where aggregator’s activities can be monitored. According to information theory, aggregation is a lossy function. A lossy function is one where the characteristics of the input data cannot be regenerated from the output of the function. An implication of this is, if in a cluster of n sensors, one of the sensors is malicious and injects false data in the system. Once data from this malicious node has been aggregated; without using any extra information, it is not possible to identify the node where this false data originated.

2. Motivation

When doing data aggregation in a sensor network, essentially we are doing a multiparty computation with the cluster head or aggregator as a trusted third party (TTP). The sensors send their sensed data to the aggregator which they trust. This aggregator, applies the required aggregate function on the data, and sends this aggregate either to the base station or to its parent. This complies exactly with the definition of MPC where multiple parties try to compute a function of their private data. In secure MPC we do away with the TTP and compute the function in network in a distributed way thus eliminating a single point of failure. As we can see data aggregation in a cluster is a case of multiparty computation. This paper tries to explore this possibility of
aggregating data in a sensor network using secure multiparty computation.

3. Secure Multiparty Computation

In multiparty computation we have a set of n players P1, P2, P3,...,Pn, holding inputs x1, x2,...xn each, who want to compute a given function f(x), without revealing their inputs to anybody. An MPC is secure if,

i. Correctness of the output is ensured even in the case of t < n corrupted players.
ii. Privacy of each player’s input is ensured.

Adversaries in secure multiparty literature have been categorized into two different types.

A semi-honest adversary is one which may try to compute more than what is required by the protocol, but does not disrupt the normal functioning. The semi honest adversary works according to the protocol providing the required output, but in addition also tries to gain extra information.

An active or malicious adversary is one which makes the machine deviate from its original program in a substantial manner. Active adversary is more dangerous because it can make the machine behave in a completely different way than what is expected.

A large body of work deals with secure multiparty computation in general, but little work has been done as far as using MPC in wireless sensor networks is concerned. Our objective here is to use SMPC to securely aggregate data in a sensor network. As discussed in section 2, data aggregation can be modeled as a multi party computation problem. Each sensor is considered as a party in an MPC scheme, and the common goal is to find out the aggregation of the data of all sensors. As is the case in MPC, the sensors don’t want to share their data with anybody else. Multiparty computation enables us to compute a function of the sensor data securely without the need of a cluster head. This helps us in solving the issues related to aggregator security. This is elaborated in the next section. An important issue in any MPC scheme is the communication complexity. The communication overhead incurred in MPC schemes is high and needs to be minimized. We look at it in more detail in section 5.

4. Aggregator Security

As can be seen in Figure 1(a), in a data aggregation scenario all the sensors in a cluster send their data to the aggregator which can also be the cluster head. In such a scenario trustworthiness of the aggregator is paramount. A corrupted aggregator under the control of an adversary can change the overall aggregated result. One method of guarding against a compromised aggregator is to have a trusted setup, as in [3] and [4]. Another method is to distribute the tasks of the aggregator in the network so that it is no more needed as shown in Figure 1(b). This approach is explored in this paper. Instead of each sensor sending its data to the cluster head for aggregation, we distribute this task in the network and make the sensors themselves compute the aggregation. Thus, we do not depend on the cluster head anymore and we do not have to worry about the adversary getting control of the cluster head. In doing so, we also eliminate issues arising from cluster head being a single point of failure.

![cluster head](a)

![To base station](b)

Figure 1 (a) Data aggregation at cluster head (b) Distributed communication.

5. Overhead in SMPC

Multiparty computation protocols traditionally have been very bulky. In an n-party situation it is largely due to all parties communicating with each other. The overhead in SMPC can be divided into two distinct parts. First, is the overhead of communication. For n parties to be able to communicate with every other party, a minimum of n*(n-1) messages need to be passed. In sensors this translates to a large amount of energy consumption, which is precisely what we are looking to minimize.

The second part is the amount of computation done at each node. This is not an issue when we have sufficient computing power on the nodes, but in the case
of sensors, amount of computation which can be done is limited by the memory capacity and the processor capability. Not a lot of research has been done in this direction.

A study of various approaches for secure MPC has brought forward the issue of striking a balance between these two kinds of overheads when designing a protocol. When we minimize the communication overhead in a protocol, the amount of computation required on each node becomes high and when we minimize the computation required on each node, the communication complexity shoots up. An SMPC scheme for sensors should take into account both these overheads and strike a balance between the two.

6. The SMPC scheme for data aggregation

We propose a protocol based on Shamir’s secret sharing scheme [5]. Using this scheme a data D can be divided into n pieces in such a way that D can be reconstructed easily from any k pieces, but no knowledge can be gained about D from k-1 or less pieces.

6.1 Preliminaries

We assume the sensors deployed in the field group themselves into clusters according to some clustering algorithm and each sensor has a unique identifier within a cluster. Each sensor is equipped with a pseudo random number generator. The base station generates a random number S which we will call the seed, and disseminates it into the network. This seed is valid only for the time period it is generated after which it expires and a new seed is disseminated periodically.

\[ n \] - number of nodes in the cluster

\[ P_i \] - i-th node of the cluster. \( 0 < i \leq n \)

\( x_i \) - data sensed by the i-th node of the cluster. \( 0 < i \leq n \)

\( k \) - number of nodes an adversary needs to compromise to break the scheme

\( h_j \) - Aggregate of the secret shares of n nodes on j-th node in the secret set.

\( S \) - Seed, or the random number disseminated by the base station.

6.2 Protocol

The seed S generated by the base station is received by all the sensors in the cluster. Using this as the input to the pseudo random number generator each sensor generates k ids. Since the input to the pseudo random number generator is same, the k ids generated by all the n sensors in the cluster are same. After the sensor has finished sensing and has some data to be sent, each node generates a random polynomial \( g_i \) of degree t-1 such that \( g_i(0) = x_i \). Each node \( P_i \) sends to all \( P_j \) the value of \( g_j(i) \), where j is the id of the node, to which the value is being sent and it takes on values from the set of k chosen ids. This is secret sharing where all the nodes in the cluster are dividing their data into k parts and sending each part to a different node. Since the seed S changes randomly after a certain period of time, it also causes the nodes to select a different set of k nodes among themselves for each time period. Let’s call this set of k nodes the secret set. These nodes in the secret set, receive one share each from every node in the cluster. After a node \( P_j \) in the secret set has received all the n shares, it aggregates them. So

\[ h_j = \sum_{k=1}^{n} g_k(j) \]

The aggregated share is then sent to each node in the secret set. When a node has received all the aggregated shares from other nodes in the secret set, it is possible for it to find out the aggregate of the original data by interpolating these k data points. The protocol is given below in a more formal manner, and an example following it.

1) The base station generates seed S and disseminates it in the network.

2) Node \( P_i \), where \( 0 < i \leq n \), generates k ids using S as input to a pseudorandom number generator. This set of k ids is called the secret set.

3) \( P_n \) generates a random polynomial \( g_i \) of degree t-1, such that \( g_i(0) = x_i \).

4) \( P_n \) calculates \( g_i(j) \), where \( j \in \text{secret set} \), and sends \( g_i(j) \) to \( P_j \).

5) Each node \( P_j \) in the secret set calculates \( h_j = \sum_{k=1}^{n} g_k(j) \) and sends the tuple \( (j, h_j) \) to all the nodes in the set.

6) Using the first element of the tuple, k Lagrange’s basis polynomials \( l_k \) are calculated, where \( k \in \text{secret set} \). Each of this polynomial is of degree k-1.

7) At each node \( H = \sum_{j}(h_j \cdot l_j) \), \( j \in \text{secret set} \), is calculated.
8) The polynomial $H$ thus obtained is equal to $\sum_{k=1}^{n} g_k$, and $H(0)$ gives us the aggregate of the data of $n$ sensors.

Our protocol needs $(n + k - 2)k$ messages to complete one round, which is an improvement over $O(n^2)$ communication complexity. An example of the working of the protocol, is given below, where $n = 4$ and $k = 3$.

After the data has been aggregated, we need a method to propagate this aggregate to the base station. We cannot let all the $k$ sensors in a cluster communicate to base station. In that case if we have $p$ clusters, it will mean $p^*k$ sensors communicating directly with the base station, which is undesirable. For this, we divide the deployment field in a number of regions as shown in the Fig. 2. Each of these regions then becomes a cluster. Aggregation starts with the region farthest from the base station. In the figure the farthest region is region $a$.

![Figure 2.: Propagating the aggregate.](image)

We assume 4 nodes $P_1$, $P_2$, $P_3$, $P_4$, with values $x_1$, $x_2$, $x_3$, $x_4$, respectively equal to 10, 20, 30 and 40. We assume $k = 3$.

Let the random polynomial generated by the 4 sensors be.

$P_1: g_1(x) = 10 + 4x + 2x^2$ \quad $g_1(1) = 16$, $g_1(2) = 26$, $g_1(3) = 40$, $g_1(4) = 58$

$P_2: g_2(x) = 20 + 3x + x^2$ \quad $g_2(1) = 24$, $g_2(2) = 30$, $g_2(3) = 38$, $g_2(4) = 48$

$P_3: g_3(x) = 30 + 2x + x^2$ \quad $g_3(1) = 33$, $g_3(2) = 38$, $g_3(3) = 45$, $g_3(4) = 54$

$P_4: g_4(x) = 40 + x^2$ \quad $g_4(1) = 41$, $g_4(2) = 44$, $g_4(3) = 49$, $g_4(4) = 56$

The sensors generate 3 random ids, Let these be 1, 3 and 4. Each sensor $i$ sends $g_i(j)$ to the 3 sensors. Sensors $P_1$, $P_2$, $P_4$, receive the shares from all the sensors, and them up.

$P_1$ receives 16, 24, 33, 41 = 114, $P_2$ and $P_4$.

$P_1$ receives 40, 38, 45, 49 = 172 $P_4$ and $P_4$.

$P_4$ receives 58, 48, 54, 56 = 216 $P_1$ and $P_5$.

Each sensor generates 3, Lagrange's basis polynomials from these tuples.

$I_0 = \left(\frac{x-2}{1-2}\right) \left(\frac{x-3}{1-3}\right) = \frac{1}{6} (x^2 - 7x + 12)$

$I_1 = \left(\frac{x-1}{2-1}\right) \left(\frac{x-4}{2-4}\right) = \frac{1}{2} (x^2 - 5x + 4)$

$I_2 = \left(\frac{x-3}{4-3}\right) \left(\frac{x-4}{4-4}\right) = \frac{1}{2} (x^2 - 4x + 3)$

$H(x) = \sum_i (I_i \cdot g_i) = -114 \left(\frac{1}{6} (x^2 - 7x + 12)\right) + 172 \left(\frac{1}{2} (x^2 - 5x + 4)\right) + 216 \left(\frac{1}{2} (x^2 - 4x + 3)\right)$

$H(0) = 100$

All the $k$ sensors will have the aggregate 100 with them.

**Example 1.**

**7. Future work**

In our research we have proposed an algorithm based on Shamir’s secret sharing scheme for securely computing the aggregation. Our current direction of work is as follows.

In [6] and [7], the authors have worked on secure MPC algorithms with linear communication complexity. The approach in their research is to divide the function to be evaluated in $n$ parts, so that each party evaluates one part. This approach although reduces the communication complexity, makes the computation intensive. This is not a concern with powerful machines, but with sensors, this is an important issue. In future we would attempt to use
the linear complexity model discussed by these authors with a scheme involving less expensive computation.

In our progress so far, we have concentrated on security issues, considering a semi-honest adversary, which tries to gather data, without disrupting the protocol. Our next step will be to consider active adversary, which tries to inject false data into the system, and make our protocol resilient to such kind of attacks.

8. Conclusions

In our work, we have proposed a secure data aggregation scheme, based on secure MPC. We use the secret sharing scheme, to divide the data generated by a sensor into n parts called shares and distribute it over the sensors in the cluster. The sensors then aggregate these secret shares and communicate with each other to calculate the final aggregation. This protocol achieves the privacy requirement of the SMC scheme. It also provides correctness in the face of a semi-honest adversary. The communication complexity of our scheme is substantially lower than $O(n^2)$.

9. Acknowledgement

This research is funded by the Intelligent Systems Center

10. References


