Modified Safe Q Protocol Based Technique for Maintaining Privacy & Integrity in Wireless Sensor Network

Patel Hardik[1]
[1]Student
[1]Department of Computer Science & Engineering
[1]Narnarayan Shastri Institute of Technology,Jetalpur

Abstract— the architecture of two-tiered sensor networks, where storage nodes serve as an intermediate tier between sensors and a sink for storing data and processing queries, has been widely adopted because of the benefits of power and storage saving for sensors as well as the efficiency of query processing. However, the importance of storage nodes also makes them attractive to attackers. SafeQ, a protocol that prevents attackers from gaining information from both sensor collected data and sink issued queries. SafeQ also allows a sink to detect compromised storage nodes when they misbehave. To preserve privacy, SafeQ uses a novel technique to encode both data and queries such that a storage node can correctly process encoded queries over encoded data without knowing their values. To preserve integrity, two schemes—one using Merkle hash trees and another using a new data structure called neighbourhood chains—to generate integrity verification information so that a sink can use this information to verify whether the result of a query contains exactly the data items that satisfy the query. To improve performance, we propose an optimization technique using Bloom filters to reduce the communication cost between sensors and storage nodes.

Key words: WSN, Storage node, Sensor, Privacy, Integrity, range query

I. INTRODUCTION

WIRELESS sensor networks (WSNs) have been widely deployed for various applications, such as environment sensing, building safety monitoring, earthquake prediction, etc. we consider a two-tiered sensor network architecture in which storage nodes gather data from nearby sensors and answer queries from the sink of the network. The storage nodes serve as an intermediate tier between the sensors and the sink for storing data and processing queries.

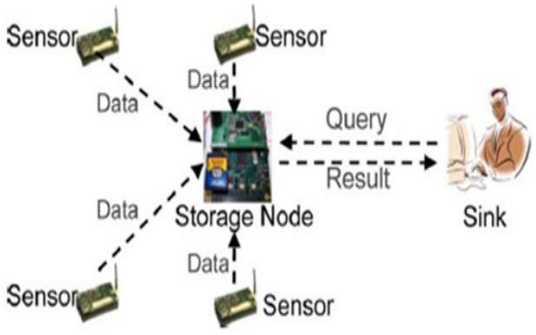


Fig. 1: Architecture of Wireless Sensor Network

Storage nodes bring three main benefits to sensor networks. First, sensors save power by sending all collected data to their closest storage node instead of sending them to the sink through long routes. Second, sensors can be memory-limited because data are mainly stored on storage nodes. Third, query processing becomes more efficient because the sink only communicates with storage nodes for queries. The inclusion of storage nodes in sensor networks was first introduced in [2] and has been widely adopted [3]–[4]. Several products of storage nodes, such as StarGate [5] and RISE [6], are commercially available. However, the inclusion of storage nodes also brings significant security challenges. As storage nodes store data received from sensors and serve as an important role for answering queries, they are more vulnerable to be compromised, especially in a hostile environment. A compromised storage node imposes significant threats to a sensor network. First, the attacker may obtain sensitive data that has been, or will be, stored in the storage node. Second, the compromised storage node may return forged data for a query. Third, this storage node may not include all data items that satisfy the query. Therefore, we want to design a protocol that prevents attackers from gaining information from both sensor collected data and sink issued queries, which typically can be modeled as range queries, and allows the sink to detect compromised storage nodes when they misbehave. For privacy, compromising a storage node should not allow the attacker to obtain the sensitive information that has been, and will be, stored in the node, as well as the queries that the storage node has received, and will receive. Note that we treat the queries from the sink as confidential because such queries may leak critical information about query issuers' interests, which need to be protected especially in military applications.

For integrity, the sink needs to detect whether a query result from a storage node includes forged data items or does not include all the data that satisfy the query. There are two key challenges in solving the privacy and integrity-preserving range query problem. First, a storage node needs to correctly process encoded queries over encoded data without knowing their actual values. Second, a sink needs to verify that the result of a query contains all the data items that satisfy the query and does not contain any forged data. SafeQ, a protocol that prevents attackers from gaining information from both sensor collected data and sink issued queries. SafeQ also allows a sink to detect compromised storage nodes when they misbehave. To preserve privacy, SafeQ uses a novel technique to encode both data and queries such that a storage node can correctly process encoded queries over encoded data without knowing their values. To preserve integrity, we propose a new data structure called a neighborhood chain that allows a sink to verify whether the result of a query contains exactly the data items that satisfy the query. In addition, we propose a solution to adapt SafeQ for event-driven sensor networks.

II. RELATED WORK

In the two-tiered sensor network, we assume that the sensors and the sink are trusted but the storage nodes are not. In a

hostile environment, both sensors and storage nodes can be compromised. If a sensor is compromised, the subsequent collected data of the sensor will be known to the attacker and the compromised sensor may send forged data to its closest storage node. It is extremely difficult to prevent such attacks without the use of tamper proof hardware. However, the data from one sensor constitute a small fraction of the collected data of the whole sensor network. Therefore, we mainly focus on the scenario where a storage node is compromised. Compromising a storage node can cause much greater damage to the sensor network than compromising a sensor. After a storage node is compromised, the large quantity of data stored on the node will be known to the attacker and upon receiving a query from the sink, the compromised storage node may return a falsified result formed by including forged data or excluding legitimate data. Therefore, attackers are more motivated to compromise storage nodes.

III. EXISTING SYSTEM

In the existing system we maintained the privacy integrity and range query which described follow is.

A. Privacy for 1-Dimensional Data:

To preserve privacy, it seems natural to have sensors encrypt data and the sink encrypt queries; however, the key challenge is how a storage node processes encrypted queries over encrypted data without knowing their actual values.

The basic idea of our solution for preserving privacy is as follows. We assume that each sensor si in a network shares a secret key ki with the sink. For the n data items d1, · · · , dn that a sensor Si collects in time slot t, Si first encrypts the data items using key ki, the results of which are represented as (d1) ki , · · · , (dn) ki . Then, Si applies a "magic" function $\mathcal{H}$ to the n data items and obtains $\mathcal{H}$(d1, · · · , dn). The message that the sensor sends to its closest storage node includes both the encrypted data and the associative information $\mathcal{H}$(d1, · · · , dn). When the sink wants to perform query {t, [a, b]} on a storage node, the sink applies another "magic" function $\mathcal{G}$ on the range [a, b] and sends {t, $\mathcal{G}$([a, b])} to the storage node. The storage node processes the query {t, $\mathcal{G}$([a, b])} over encrypted data (d1) ki , · · · , (dn) ki collected at time slot t using another "magic" function $\mathcal{F}$.

The three "magic" functions $\mathcal{H}$, $\mathcal{G}$, and $\mathcal{F}$. Satisfy the following three conditions:

(1) A data item dj ($1 \le j \le n$) is in range [a, b] if and only if $\mathcal{F}$ (j, $\mathcal{H}$(d1, · · , dn), $\mathcal{G}$([a, b])) is true.This condition allows the storage node to decide whether (dj)ki should be included in the query result.

(2) Given, $\mathcal{H}$(d1, · · · , dn) and (dj) ki , it is computationally infeasible for the storage node to compute dj. This condition guarantees data privacy.

(3) Given $\mathcal{G}$([a, b]), it is computationally infeasible for the storage node to compute [a, b]. This condition guarantees query privacy. Figure 2[1] illustrates the above basic idea.

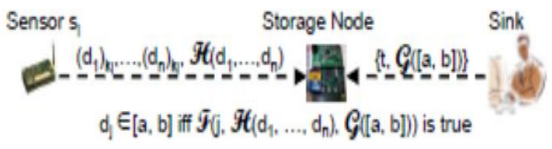


Fig. 2: Safe Q for preserving privacy

B. Algorithm to Provide Confidentiality:

- Split the data into n data blocks d1, d2, …, dn where n=5
- Encrypt the data using secret key E(Si,di) where i=5
- Let each of the n data block contain m bytes of data d1, d2, …, dm where m=1024
- For each data block sent, generate a hash code using SHA1, MD5

The data sent to storage node contains the following information:

Sensor Storage node: Sid, Si, {d1, d2, …,d5}, HMAC { d1, d2, …, d5} Where, Sid is the Sensor ID, Si is the Secret Key. Confidentiality in storage node is ensured by providing access only to authorized users through the secret key shared between sensor and storage node (as shown in fig 3)[7].

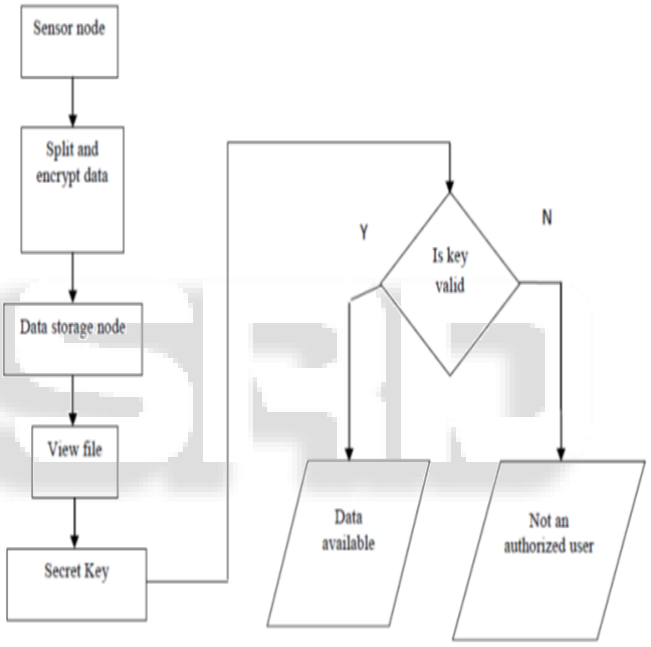


Fig. 3: flowchart to show confidentiality

C. Integrity for 1-Dimensional Data:

The meaning of data integrity is two-fold in this context. In the result that a storage node sends to the sink in responding to a query, first, the storage node cannot include any data item that does not satisfy the query; second, the storage node cannot exclude any data item that satisfies the query. To allow the sink to verify the integrity of a query result, the query response from a storage node to the sink consists of two parts:

(1) The query result QR, which includes all the encrypted data items that satisfy the query;

(2) The verification objects VO, which includes information for the sink to verify the integrity of QR.

We first present a new data structure called neighborhood chains and then discuss its use in integrity verification. Given n data items d1, · · · , dn, where d0 < d1 < · · · < dn < dn+1, we call the list of n items encrypted using key ki, (d0|d1)ki , (d1|d2)ki , · · · , (dn−1|dn)ki , (dn|dn+1)ki, the neighborhood chain for the n data items. Here "|" denotes concatenation. For any item (d j−1|dj)ki in

the chain, we call dj the value of the item and (dj|dj+1)ki the right neighbor of the item. Figure 4[1] shows the neighborhood chain for the 5 data items 1, 3, 5, 7 and 9.

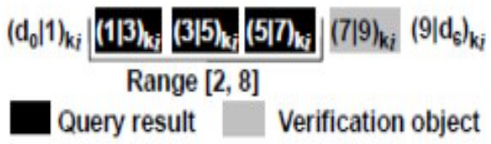


Fig. 4: Neighborhood chain

Preserving query result integrity using neighborhood chaining works as follows. After collecting n data items d1, · · · , dn, sensor Si sends the corresponding neighborhood chain (d0|d1)ki , (d1|d2)ki , · · · , (dn−1|dn)ki , (dn|d n+1)ki, instead of (d1)ki , · · ·, (dn)ki , to a storage node. Given a range query [a, b], the storage node computes QR as usual. The corresponding verification object VO only consists of the right neighbor of the largest data item in QR. Note that VO always consists of one item for any query. If QR = {(dn1−1|dn1)ki , · · · , (d n2−1|dn2)ki }, then VO ={(dn2 |dn2+1)ki }; if QR = Ø, suppose dn2 < a ≤ b < dn2+1, then VO = {(dn2 |dn2+1)ki }.

After the sink receives QR and VO, it verifies the integrity of QR as follows. First, the sink verifies that every item in QR satisfies the query. Second, the sink verifies that the storage node has not excluded any item that satisfies the query. Let {(dn1−1|dn1)ki , · · ·, (dj−1|dj)ki, · · · , (dn2−1|dn2)ki } be the correct query result and QR be the query result from the storage node. We consider the following four cases.

If there exists n1 < j < n2 such that (dj−1|dj)ki ∉ QR, the sink can detect this error because the items in QR do not form a neighborhood chain.

If (dn1−1|dn1)ki ∉QR, the sink can detect this error because it knows the existence of dn1 from (dn1 |dn1+1) ki and dn1 satisfies the query.

If (dn2−1|dn2)ki ∉QR, the sink can detect this error because it knows the existence of dn2 from the item (dn2 |dn2+1)ki in VO and dn2 satisfies the query.

If QR =Ø, the sink can verify this fact because the item (dn2 |dn2+1)ki in VO should satisfy the property dn2 < a ≤ b < dn2+1.

Note that our submission and query protocols are designed to facilitate integrity verification. To process query [a, b] over data items d1, · · · , dn, instead of testing whether each data item di is in [a, b], we test which ranges among [d0, d1], [d1, d2], · · ·, [dn, dn+1] contain a and which ranges contain b. Thus, a storage node not only can find which items satisfy a query, but also can find the right neighbor of the largest data item in the query result, which is the verification object.

D. Algorithm to Provide Integrity:

- The HMAC values of the data blocks sent is stored in the sensor.
- The sensor can verify for integrity of each data block it has sent.
- Compare the HMAC value of the data block in storage node with HMAC value of data block in sensor.
- If the values are same there is no modification in data; if values are different indicates data modification.
- Verification of block d1 for eg:
- If HMAC (d1) at sensor = HMAC(d1) at storage node => No data modification.
- If HMAC (d1) at sensor != HMAC(d1) at storage node => Data modification.

A forged message is detected (as shown in fig 5) if a parent's calculated HMAC is inconsistent with HMAC received from the child.[7]

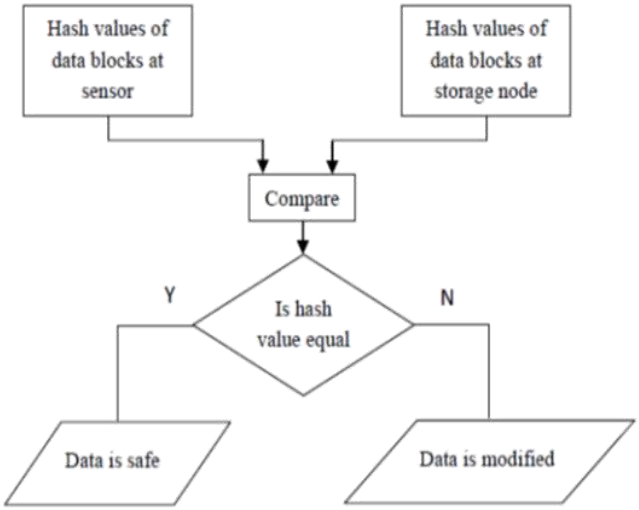


Fig. 5: Flowchart to Show Integrity

E. Range Queries in Sensor Network:

In this paper, we address the above challenge by sensors reporting their idle period to storage node each time when they submit data after an idle period or when the idle period is longer than a threshold. Storage nodes can use such idle period reported by sensors to prove to the sink that a sensor did not submit any data at any time slot in that idle period.

Next, we discuss the operations carried on sensors, storage nodes and the sink.

F. Sensors:

An idle period for a sensor is a time slot interval [t1, t2], which indicates that the sensor has no data to submit from t1 to t2, including t1 and t2. Let γ be the threshold of a sensor being idle without reporting to a storage node. Suppose the last time that sensor Si submitted data or reported idle period is time slot t1 − 1. At any time slot t ≥ t1, Si acts based on the following three cases:

(1) t = t1: In this case, if Si has data to submit, then it just submits the data; otherwise it takes no action.

(2) t1 < t < γ + t1 − 1: In this case, if si has data to submit, then it submits data along with encrypted idle period [t1, t−1]ki; otherwise it takes no action. We call [t1, t − 1]ki an idle proof.

(3) t = γ + t1 − 1: In this case, if Si has data to submit, then it submits data along with the idle proof [t1, t−1]ki; otherwise, it submits the idle proof [t1, t]ki .

Figure 6 illustrates some idle periods for sensor Si, where each unit in the time axis is a time slot, a grey unit denotes that Si has data to submit at that time slot, and a blank unit denotes that Si has no data to submit at that time slot. According to the second case, at time slot t2+1, Si

submits data along with the idle proof [t1, t2]ki . According to the third case, at time slot t4, Si submits the idle proof [t3, t4]ki .

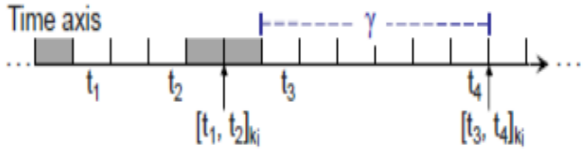


Fig. 6: idle periods and data submissions

G. Storage Nodes:

When a storage node receives a query {t, $\mathcal{G}$([a, b])} from the sink, it first checks whether Si has submitted data at time slot t. If Si has, then the storage node sends the query result. Otherwise, the storage node checks whether Si has submitted an idle proof for an idle period containing time slot t. If true, then it sends the idle proof to the sink as VO. Otherwise, it replies to the sink saying that it does not have the idle proof containing time slot t at this moment, but once the right idle proof is received, it will forward to the sink. The maximum number of time slots that the sink may need to wait for the right idle proof is γ−1. Here γ is a system parameter trading off efficiency and the amount of time that sinks may have to wait for verifying data integrity. Smaller γ favors the sink for integrity verification and larger γ favors sensors for power saving because of less communication cost.

H. The Sink:

Changes on the sink side are minimal. In the case that VO lacks the idle proof for verifying the integrity of QR, it will defer the verification for at most γ −1 time slots, during which benign storage nodes are guaranteed to send the needed idle proof.

IV. Proposed Method

We propose SafeQ, a protocol that prevents attackers from gaining information from both sensor collected data and sink issued queries. SafeQ also allows a sink to detect compromised storage nodes when they misbehave
The important parts in this protocol are:

- Storage node
- Sensor
- Sink

To preventing from attackers and compromised node we define:

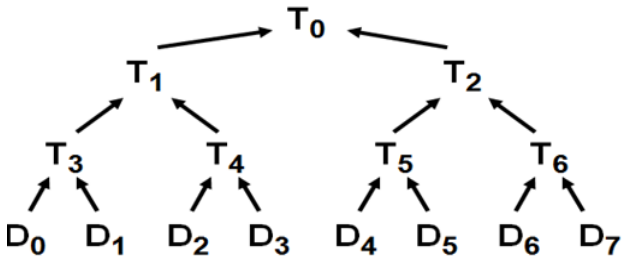
- Merkle Hash tree for privacy
- Micro Hash chaining for integrity

A. Merkle Hash Tree:

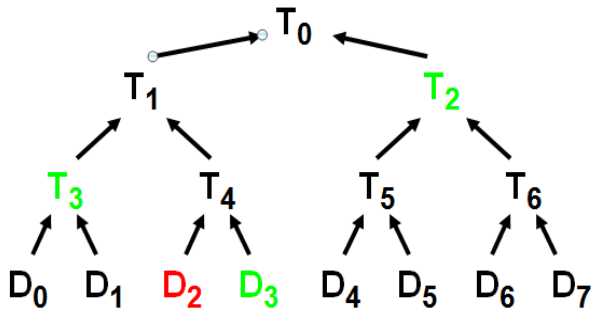
Hash trees can be used to verify any kind of data stored, handled and transferred in and between computers. Currently the main use of hash trees is to make sure that data blocks received from other peers in a peer-to-peer network are received undamaged and unaltered, and even to check that the other peers do not lie and send fake blocks.

B. Merkle Hash Tree Work in Three Steps:

- Key generation
- Signature generation
- Signature verification
- Authenticate a sequence of data values D0 , D1 , …, DN
- Construct binary tree over data values



- Verifier knows T0
- How can verifier authenticate leaf Di ?
- Solution: recomput T0 using Di
- Example authenticate D2 , send D3 T3 T2
- Verify T0 = H(H(T3 || H(D2 || D3)) || T2)



1) Advantages of Proposed System:

- The system is privacy- and integrity- preserving.
- SafeQ significantly strengthens the security of two-tiered sensor networks.
- In terms of efficiency, our results show that SafeQ significantly outperforms prior art for multidimensional data in terms of both power consumption and storage space.

V. Conclusion

We proposes SafeQ a novel and efficient protocol for handling range queries in two tiere sensor networks in a privacy and integrity preserving fashion. SafeQ uses the techniques of prefix membership verification Merkle hash trees and neighborhood chaining. In terms of security SafeQ significant strength is the security of two tiere sensor networks. In the prior art SafeQ prevents a compromising storage node from obtains reasonable estimation on the actual values of sensor collecte data items and sink issue queries.

References

[1] F. Chen and A. X. Liu, “SafeQ: Secure and efficient query processing in sensor networks,” in Proc.IEEE INFOCOM, 2010.
[2] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, “Data-centric storage in sensornets with GHT, a geographic hash table,” Mobile Netw. Appl., vol. 8, no. 4, pp. 427–442, 2003.
[3] P. Desnoyers, D. Ganesan, H. Li, and P. Shenoy, “Presto:A predictive storage architecture for sensor networks,” in Proc. HotOS, 2005, p. 23.

[4] B. Sheng and Q. Li, "Verifiable privacy-preserving range query in wotiered sensor networks," in Proc. IEEE INFOCOM, 2008, pp. 46–50.
[5] Xbow, "Stargate gateway (spb400)," 2011 [Online]. Available: http:// www.xbow.com
[6] W. A. Najjar, A. Banerjee, and A. Mitra, "RISE:More powerful, energy efficient, gigabyte scale storage high performance sensors," 2005 [Online]. Available: http://www.cs.ucr.edu/~rise
[7] Kusuma K V,Prasad M R, " Confidential and Reliable Data Storage in WSN" in Proc.ijarcsse,4, April 2013.
[8] H. Krawczyk, M. Bellare, and R. Canetti, "Hmac: Keyed-hashing for message authentication," RFC 2104, 1997.
[9] R. Zhang, J. Shi, and Y. Zhang, "Secure multidimensional range queries in sensor networks," in Proc. ACM MobiHoc, 2009, pp. 197–206.
[10] D. Zeinalipour-Yazti, S. Lin, V. Kalogeraki, D. Gunopulos, and W. A. Najjar, "Microhash: An efficient index structure for flash-based sensor devices," in Proc. FAST, 2005, pp. 31–44.
[11] B. Sheng, Q. Li, and W. Mao, "Data storage placement in sensor networks," in Proc. ACM MobiHoc, 2006, pp. 344–355.

