Ensuring Data Storage Security against Frequency-Based Attacks in Wireless Networks

Hongbo Liu[1], Hui Wang[2], and Yingying Chen[1]

[1] Dept. of ECE, Stevens Institute of Technology
Castle Point On Hudson, Hoboken, NJ, 07030, USA
{hliu3,yingying.chen}@stevens.edu
[2] Dept. of Computer Science, Stevens Institute of Technology
Castle Point On Hudson, Hoboken, NJ, 07030, USA
hwang@cs.stevens.edu

Abstract. As wireless networks become more pervasive, the amount of the wireless data is rapidly increasing. One of the biggest challenges is how to store these data. To address this challenge, distributed data storage in wireless networks has attracted much attention recently, as it has major advantages over centralized approaches. To support the widespread adoption of distributed data storage, secure data storage must be achieved. In this work, we study the frequency-based attack, a type of attack that is different from previously well-studied ones, that exploits additional adversary knowledge to crack the encrypted data. To cope with frequency-based attacks, the straightforward 1-to-1 substitution encryption functions are not sufficient. We propose a data encryption strategy based on 1-to-n substitution via dividing and emulating techniques such that an attacker cannot derive the mapping relationship between the encrypted data and the original data based on their knowledge of domain values and their occurrence frequency. Our simulation results show that our data encryption strategy can achieve high security guarantee with low overhead.

1 Introduction

As the rapid advancement of wireless technologies has led to a future where wireless networks are becoming a part of our social life, the collected wireless data provides tremendous opportunities to support various applications ranging from environmental sensing, to infrastructure monitoring, to mobile social network analysis. However, as the amount of the wireless data is increasing, one of the biggest challenges in wireless networks is how to store these data. There are two possible ways: centralized and distributed. The traditional approach is to store the collected wireless data in a centralized manner. For example, in wireless sensor networks (WSNs) the sensing data is collected from each individual sensor and sent back to a central server for data access. However, the centralized approaches may result in performance bottlenecks of data access, and a single point of failure to both server compromise and intentional attacks.

To address these problems, distributed data storage [1,2,3,4,5] in wireless networks recently have attracted much attention. For instance, the sensed data can be stored by its type at a group of storage nodes in the network to perform data-centric storage or stored at each individual device that collects the data. The distributed data storage has major advantages over centralized approaches: storing the data on the collected wireless devices or in-network storage nodes decreases the need of constant data forwarding back to centralized places, which largely reduces the communication in the network and the energy consumption on individual devices, and consequently eliminates the existence of centralized storage and enables efficient and resilient data access. Furthermore, as wireless networks become more pervasive, new-generation wireless devices with significant memory enhancement and powerful processing capabilities are available (e.g., smart phones and laptops), making the deployment of distributed data storage not only feasible but also practical.

However, secure data storage must be achieved before widespread adoption of distributed data storage. Prior work in wireless network security has been focused on network communication security such as key management, secure localization, and intrusion detection [6, 7, 8, 9, 10]. None of these works have addressed the problem of secure distributed data storage. To fulfill the security requirements raised by the distributed data storage, recent research has started studying distributed access control, data confidentiality, and data integrity. [11] introduced a redundancy-based key distribution scheme that utilizes secret sharing to achieve a decentralized certificate authority. [12] studied to perform secure distributed data storage by developing an adaptive polynomial-based data storage scheme. [13] presented a dynamic data integrity checking scheme for verifying the consistency of data shares in a distributed manner, which is constructed based on the principle of algebraic signatures to ensure the integrity of data shares.

Most of these current research aim to provide data confidentiality, dependability, and integrity from the perspective that the adversaries will make efforts to access the data by cracking the data encryption mechanisms with little prior knowledge. None of these studies have investigated the problem of attackers cracking the data encryption by exploiting additional adversary knowledge. In particular, today with rapidly evolving adversarial activities, an attacker may possess prior knowledge about the domain values and even the exact occurrence frequencies of the data values. For instance, for the distributed data storage that stores the coordinate values of locations that people visited, the attacker may know: (1) where are the most popular locations, and (2) the fact that the frequency of these locations should be higher than that of all the other locations. Then those encrypted data values of the highest frequency must map to these popular locations. The problem gets even worse when we consider a more conservative model that the attacker knows the *exact* frequency of some or all original data values and utilizes such knowledge to crack the data encryption by matching the encrypted data values with original data values based on their frequency distribution. We call this kind of attack as *frequency-based attack*. Frequency-based attacks are especially harmful in distributed data storage. For instance, an attacker can derive the specific activities of an important officer if the attacker knows the frequency of his visited places, or a hunter can wait at specific locations of an endangered animal by possessing the knowledge of the frequency of the animal's habitation-related movements.

To cope with frequency-based attacks, apparently 1-to-1 substitution encryption function is not enough. A stronger mechanism is needed to provide more reliable data protection. However, little work has been explored to cope with frequency-based attacks. [14] has developed a secure encryption scheme to mitigate frequency-based attacks in centralized database, making it not applicable to support secure distributed data storage in wireless networks. In this paper, we propose a data encryption strategy based on 1-to-n substitution to defend against frequency-based attacks on distributed data storage in wireless networks. Our data encryption strategy aims to transform the original frequency distribution of the original data (i.e., plaintext) to a uniform distribution for the encrypted data (i.e., ciphertext) so that the attacker cannot derive the mapping relationship between the encrypted data and the original data based on their knowledge of domain values and their occurrence frequency.

In particular, we develop two techniques, *dividing* and *emulating*, to achieve the uniform distribution of encrypted data either on an individual device or across the network to cope with two types of attackers: *global frequency-based attack*, whereby the attacker only has the knowledge of the global occurrence of the data in the network, and *local frequency-based attack*, whereby the attacker's knowledge is advanced knowing about the specific occurrence frequency of the data on each individual device.

As the data is encrypted in the network for security purpose, another important issue is how to efficiently evaluate queries over these encrypted data. A naive method is to transfer all encrypted data in the network to the trusted nodes for decryption and query evaluation, which will incur tremendous communication overhead. Thus we design an efficient query evaluation procedure based on the dividing and emulating techniques for both types of point and range queries that are representative to support real-time data queries in wireless networks. Both the theoretical analysis and simulation results show that our 1-to-n substitution data encryption strategy can achieve high security guarantee, low computational cost, and efficient query processing.

The remainder of the paper is organized as follows. In Section 2, we first set up the network model, describe the attack model, and provide an overview of our strategy. We then describe our 1-to-n substitution data encryption strategy and present the details of the query evaluation procedure in Section 3. In Section 4 we discuss our simulation methodology, evaluation metrics, and results that validate our approach. Finally, we conclude in Section 5.

2 System Overview

2.1 Network Model

In our system, we consider wireless networks consisting of both static and mobile nodes, where each node represents a wireless device that can take the form of sensor, active RFID tag, laptop, or smart phone. We assume the collected data will be stored within the network at each node unless it is required to be sent to a centralized storage space for backup. By uploading data in a lazy fashion (i.e., on-demand only), distributed data storage enables real-time query evaluation and avoids frequent data transfer from the wireless devices to the centralized storage, and consequently reduces massive battery power consumption and vastly decreases the communication overhead of the network.

To prevent the misuse of the data and provide the confidentiality of the data, the data is encrypted in our network. We refer the original unencrypted data values as *plaintext* and the encrypted values as *ciphertext*. When a user queries the data in the network, all the nodes holding the data that matches the query will respond to the user by sending back the corresponding ciphertext. The user is responsible to perform the data decryption.

2.2 Attack Model

In this section, we first provide an example of frequency-based attacks. We then categorize the knowledge and behavior of the adversaries.

Example. We assume there is a set of data collected by the wireless devices where it represents the location information of animals usually appearing as shown in Table 1 (a). The original data contains the animal type and the location information.

Table 1. Data example: (a) the original data table; (b) after 1-to-1 encryption; and (c) after 1-to-n substitution encryption via dividing and emulating

Animal Type	**Location**
Panda	river A
Panda	river A
deer	wood B
Panda	wood C
deer	wood B
Panda	river A

(a)

Animal Type	**Location**
123	river A
123	river A
128	wood B
123	wood C
128	wood B
123	river A

(b)

Animal Type	**Location**
123	river A
123	river A
128	wood B
125	wood C
128	wood B
125	river A

(c)

As the panda is an endangered species, the information about panda's activities is sensitive and should be protected to avoid the access by poachers. The straightforward way is to use 1-to-1 encryption function to encrypt the animal type in the data set (as in Table 1 (b)). However, if a poacher has the knowledge of the animals' occurrence frequency in the dataset, there are 4 panda's record entries and 2 deer's record entries as depicted in Table 1 (a), the poacher can map the occurrence frequency of the encrypted data (e.g., 4 times of 123) to derive the corresponding animal type (e.g., panda) without decrypting the data, and consequently access the location information. In this case, the poacher will gain the sensitive information that the panda often appears at river A.

Based on the knowledge level of the original data's occurrence frequency that an adversary has, in this work we categorize the frequency-based attacks into two types: *global* and *local*.

Global Frequency-based Attack. An adversary only has the knowledge of the overall distribution of the data in the network. In particular, the adversary knows the occurrence frequency of a plaintext PT_j as $f_j = \sum_{i=1}^{N} f_{j,i}$, where $i = 1, 2, \cdots, N$ and N is the number of nodes in the network. Thus, the occurrence frequency of all the plaintext $freq(PT)$ in the network can be expressed as: $freq(PT) = \sum_{j=1}^{k} f_j$, with $j = 1, 2, \cdots, k$ and k is the number of distinctive plaintext values. However, the adversary does not have the knowledge of the detailed occurrence frequency of the data on each individual wireless device.
Local Frequency-based Attack. An adversary has the advanced knowledge of the distribution of plaintext values on each individual wireless device. Particularly, the adversary knows the occurrence frequency of a plaintext PT_j as $f_{j,i}$, with $j = 1, 2, \cdots, k$ and $i = 1, 2, \cdots, N$. k is the number of distinctive plaintext values and N is the number of nodes in the network. The local frequency-based attacks are more harmful as the attacker can derive the mapping between the encrypted data and the original data on each individual device independently.

2.3 Approach Overview

Encrypting Data via Dividing and Emulating. Based on the simple example in Table 1, we showed that simply encrypting the original sensitive data values that we want to protect by using 1-to-1 encryption functions and storing the ciphertext will result in encrypted values following the same distribution as the original plaintext values, making it easy to launch frequency-based attacks and disclosing the sensitive data to adversaries. To cope with frequency-based attacks, we propose to *divide* each plaintext value into one or more ciphertext values in such a way that regardless of the original data distribution, the target distribution remains close to flat, i.e., uniform. Furthermore, we propose *emulating* on the divided data to fit the target distribution to a uniform distribution, so that the attacker cannot uniquely crack the identity of ciphertext values, i.e., deriving the corresponding plaintext values, based on his knowledge of data frequency. Table 1 (c) shows that after applying dividing and emulating techniques, the distribution of the ciphertext values is uniform, i.e., 2 times for 123, 125, and 128 each, highly decreasing the probability for an adversary to derive the plaintext values by launching a frequency-based attack.
Coping with Attacks. Under global frequency-based attacks, our dividing and emulating techniques will exploit the global frequency distribution of plaintext values in the network to achieve uniform frequency distribution of the ciphertext values in the whole network, i.e., by examining the distribution of the encrypted values, the frequency of each data value in the network will be nearly uniform for the attacker. Whereas under local frequency-based attacks, the uniform distribution of the target ciphertext will be achieved on individual wireless device independently. Thus, the occurrence frequency of ciphertext on different nodes may be different.
Answering Data Query. We consider two types of queries on the data: *point queries* that return all data values in the network that equal to a given value, and *range queries* that return all data values in the network that fit in a range.

The query processing consists of three phases: (1) query translation at user side that transforms the original queries containing plaintext values to the ones with corresponding ciphertext values, (2) query evaluation at nodes in network that issues the translated queries on the encrypted data. For both point and range queries, since the ciphertext values are encrypted by order preserving encryption function, the ciphertext values whose plaintext values matching the original query will be returned, and (3) query post-processing at user side that decrypts the returned ciphertext values to plaintext values as the answer of the original queries.

3 Dividing and Emulating: 1-to-n Substitution Encryption

In this section, we first describe our *dividing* and *emulating* techniques that are used in 1-to-n Substitution Encryption. We then present our efficient query processing over encrypted data by using dividing and emulating techniques.

3.1 Dividing

The basic idea of dividing is that, any plaintext value of frequency f is divided into multiple ciphertext values such that the total frequency of these ciphertext values equals to f. Intuitively, if k unique plaintext values are split into $m > k$ unique ciphertext values that are of the same frequency, none of these ciphertext values can be explicitly mapped to their corresponding plaintext values by the frequency-based attack. Indeed, the *decipher probability* P that these m ciphertext values can be correctly mapped to k plaintext values by the frequency-based attack equals

$$P = \frac{1}{\binom{m-1}{k-1}}. \tag{1}$$

Number of Divided Ciphertext Values. To achieve a threshold σ of the decipher probability, for k unique plaintext values that are encrypted into m unique ciphertext values of the same frequency, they must satisfy $P = \frac{1}{\binom{m-1}{k-1}} \leq \sigma$. Intuitively, the smaller σ is, the more robust the dividing scheme is when against the frequency-based attack. Our goal is to calculate the appropriate value of m (i.e., the number of unique ciphertext values), with given σ and k. However, directly deriving m from the constraint $\frac{1}{\binom{m-1}{k-1}} \leq \sigma$ is computationally hard. Thus we consider Stirling's approximation, i.e., $m! \approx m^m e^{-m}\sqrt{2\pi m}$. We thus have:

$$P = \frac{1}{\binom{m-1}{k-1}} = \frac{1}{\frac{(m-1)!}{(k-1)!(m-k)!}} \approx \frac{1}{\frac{(m-1)^{(m-1)}\sqrt{2\pi(m-1)}}{2\pi(k-1)^{(k-1)}(m-k)^{(m-k)}\sqrt{(k-1)(m-k)}}}$$

$$\leq \frac{1}{\frac{(m-1)^{(m-1)}\sqrt{m-1}}{\sqrt{2\pi}((k-1)^{(k-1)}(m-1)^{(m-k)}\sqrt{(k-1)(m-1)})}} = (\frac{k-1}{m-1})^{(k-1)}\sqrt{2\pi(k-1)}. \tag{2}$$

As it is required that $P = \frac{1}{\binom{m-1}{k-1}} \leq \sigma$, from Equation 2, we can infer that $m \geq (k-1)(\frac{\sqrt{2\pi(k-1)}}{\sigma})^{\frac{1}{k-1}} + 1$. It is straightforward that larger m value implies the more robustness of the dividing scheme against the frequency-based attack. However, as we will discuss soon, larger m values will also result in more keys needed for encryption and decryption. To balance the trade-off between the robustness of the scheme and the cost for key management, we use

$$m = (k-1)(\frac{\sqrt{2\pi(k-1)}}{\sigma})^{\frac{1}{k-1}} + 1 \qquad (3)$$

as the number of divided ciphertext values needed to achieve the required robustness of the scheme. Based on the valued m, next, we discuss how to split k unique plaintext values into m unique ciphertext values.

Dividing Factor. We define the *dividing factor* in our dividing scheme as following.

Definition 1. *Given k unique plaintext values $PT_j (1 \leq j \leq k)$ that will be encrypted as m unique ciphertext values, let $f = \Sigma_{j=1}^{k} f_j$, where f_j is the frequency of the plaintext value PT_j. Then each PT_j is encrypted as $\lceil \frac{f_j}{d} \rceil$ unique ciphertext values, where*

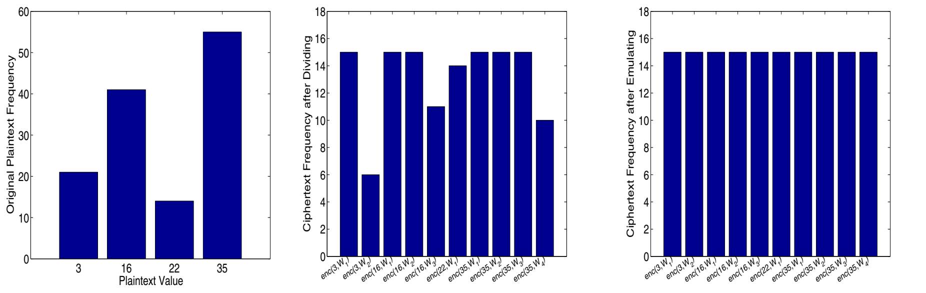
$$d = \lceil \frac{f}{m} \rceil. \qquad (4)$$

We call d the dividing factor*.*

After dividing, k unique plaintext values are split into m unique ciphertext values, such that $m - \lceil \frac{md-f}{d} \rceil$ of them are of the same frequency d. If $md = f$, then all m ciphertext values are of the same frequency. Otherwise, out of these m values, there will be $\lceil \frac{md-f}{d} \rceil$ of them with frequency of $f_j - \lfloor \frac{f_j}{d} \rfloor \times d$, where f_j is the frequency of their corresponding plaintext values.

Dividing Procedure. Next, we describe the details of our dividing procedure that can achieve the goal mentioned above.
Step I. Sorting: We sort the plaintext values by their frequencies in ascending order. Let $\delta = min(PT_{j+1} - PT_j)(1 \leq j \leq k-1)$ be the minimal interval between any two successive frequency values.
Step II. Dividing: For each plaintext value PT_j, we choose t distinct random numbers $w_1, \ldots, w_t (1 \leq t \leq \lceil \frac{f_j}{d} \rceil)$ as weight values, where f_j is the frequency of PT_j, and d is the dividing factor. We require that w_2 should be unique among all the w_is, as it is needed for value decryption (More details are in Section 3.3). Then we partition f_j number of PT_j values into $\lceil \frac{f_j}{d} \rceil$ partitions, each partition containing d number of PT_j values, except the last one that contains $f_j - \lfloor \frac{f_j}{d} \rfloor \times d$ number of PT_j values. Then the PT_j value in the i-th partition $(1 \leq i \leq \lceil \frac{f_j}{d} \rceil)$ is encrypted to

$$CT_i = enc(PT_j + \sum w_i \delta), 1 \leq i \leq \lceil \frac{f_j}{d} \rceil, \qquad (5)$$



(a)Original plaintext freq. (b)Ciphertext freq. after dividing (c)Ciphertext freq. after emulating

Fig. 1. Dividing and Emulating

where w_i is a distinct random number $\in (0, 1/(\lceil \frac{f_j}{d} \rceil + 1))$ (i.e., $\Sigma w_i < 1$), and *enc*() is an order-preserving encryption function [15]. More specifically, the first partition of occurrence of PT_j will be transformed to $enc(PT_j + w_1\delta)$; the l-th partition of occurrences to $enc(PT_j + \sum_{1 \le i \le l}(w_i\delta))$. That is to say, the l-th partition is displaced from PT_j by a fraction of the gap δ given by the sum $w_1 + w_2 + \cdots + w_l$. After dividing, there are $\lceil \frac{f_j}{d} \rceil$ number of ciphertext values $CT_1, \ldots, CT_t (1 \le t \le \lceil \frac{f_j}{d} \rceil)$, with their total frequencies equal to f_j.

To illustrate the results of ciphertext values by applying our dividing technique, we show a simple example as following: Given two plaintext values PT_1 and PT_2 of frequency 12 and 21, $f = 12 + 21 = 33$. Assume Equation 3 has returned $m = 5$. Then using Definition 1, the dividing factor d is calculated as 7. Based on the dividing procedure, PT_1 will be encrypted as 2 unique ciphertext values, one of frequency 7, and one of frequency 5, by using 2 unique keys; PT_2 will be encrypted as 3 unique ciphertext values, each of frequency 7.

Due to the use of order-preserving encryption function *enc*(), a nice property of the dividing scheme is that the ciphertext corresponding to different plaintext values will not straddle each other. More precisely, for any two values $PT_i < PT_j$, and for any ciphertext values CT_i^m,CT_j^n (i.e., the m-th and n-th ciphertext values of PT_i and PT_j respectively), it is necessary that $CT_i^m \le CT_j^n$. This will enable the efficient query evaluation over the ciphertext values (More details of query evaluation will be discussed in Section 3.3).

Cost of Key Management. For each plaintext value that is divided into r unique ciphertext values, we need r unique keys. To reduce the total number of keys that are needed for dividing k unique plaintext values in the network, we allow these plaintext values share keys for dividing. Therefore, the number of keys r needed for the dividing scheme equals to $r = max_{1 \le j \le k} \lceil \frac{f_j}{d} \rceil$, which largely reduces the total number of unique keys during encryption.

3.2 Emulating

The dividing procedure cannot guarantee that all ciphertext values are of the same frequency. Figure 1 (a) and (b) depict an example of dividing. The 4 plaintext values 3, 16, 22, and 35 of occurrence frequency 21, 41, 14 and 55

(Figure 1 (a)) are divided into 10 unique ciphertext values (Figure 1 (b)), with the dividing factor as 15. Figure 1 (b) shows some ciphertext values, including the second ciphertext value of plaintext value 3, the third ciphertext value of plaintext value 16, the first ciphertext value of plaintext value 22, and the last ciphertext value of plaintext value 35, that are of different frequency from the other ciphertext values. These ciphertext values may face the threat that their encryption can be cracked by the frequency-based attack.

Thus, we apply *emulating* on these values, so that these ciphertext values are indistinguishable from the others by their frequencies. In particular, for these ciphertext values, they are duplicated so that their frequency also equals to d, the frequency of the other ciphertext values. Figure 1 (c) shows the results of the frequencies of these ciphertext values after emulating. Therefore, By performing emulating, these ciphertext values are indistinguishable by the frequency-based attack. However, it incurs additional space overhead for the duplicates, which is called emulating noise. There exists a trade-off between the security guarantee and the space overhead i.e., higher security guarantee may lead to more space overhead. This trade-off will be studied in details in Section 4.

To cope with both global and local frequency-based attacks, we apply the dividing and emulating encryption scheme on the plaintext values. For global frequency-based attacks, we apply the scheme on the global distribution information to achieve globally uniform frequency distribution of the ciphertext values (i.e., all unique ciphertext values in the network are of the same frequency). While for local frequency-based attacks, we exploit the dividing and emulating techniques locally on each individual device, so that the ciphertext values on each device will achieve uniform frequency distribution.

3.3 Efficient Query Processing over Encrypted Data

We assume the users issue their queries that only contain plaintext values. In this paper, we consider two types of queries: *point queries* that return all data values in the network that equal to a given value, and *range queries* that return all data values in the network that fit in a range $[l, u]$. Our goal is to translate the plaintext queries to ciphertext queries that can be applied directly on the encrypted data in the network. This mechanism has two advantages: (1) sending ciphertext queries to the network will protect the queries, especially the plaintext values in the queries, from the malicious attackers, and (2) it supports efficient query evaluation, as data decryption in the network, which in general is costly, is avoided. To achieve the goal, we design the query processing procedure that consists of three phases: query translation at the user side, query evaluation at the nodes in the network, and query post-processing at the user side. Next, we discuss the details of these three phases.

Phase-1: Query translation at user side. We assume a user can access all the auxiliary information including the weight values w_i, the gap values δ, and the order-preserving encryption function $enc()$ that are used in the dividing scheme. He/she will make use of these information to translate the plaintext queries as following.

Point queries: Given a point query $Q : V = v$, the user will translate it to Q' by following the same dividing scheme for encrypting data values in the network. In particular, the plaintext value v will be encrypted to r ciphertext values $CT_1, \ldots, CT_r$. Since these r ciphertext values follow the order that $CT_1 < CT_2 \cdots < CT_r$, the query Q will be translated to $Q' : V \in [CT_1, CT_r]$, where $CT_1 = enc(v + w_1 * \delta)$, and $CT_r = enc(v + \sum_{i=1}^{r} w_i * \delta)$. Here w_i and δ are pre-valued in the dividing scheme (see Section 3.1).

Range queries: Recall that our dividing technique performs order-preserving encryption. Thus the range query $Q : V \in [l, u]$ will be translated to another range query Q'. In particular, let w_i^l and w_i^u be the i-th weight values assigned for dividing l and u, and δ_l, δ_u be the gap values used for dividing l and u values, then the query Q will be translated to $Q' : V \in [CT_1^l, CT_r^u]$, where $CT_1^l = enc(l + w_1^l * \delta_l)$, and $CT_r^u = enc(u + \sum_{i=1}^{r} w_i^u * \delta_u)$. In other words, the plaintext range $[l, u]$ is translated to another range whose lower bound equals to the smallest divided ciphertext value of l, and upper bound equals to the largest divided ciphertext value of u.

Phase-2: Query evaluation at nodes in the network. After translation, the range query $Q' : V \in [CT_l, CT_u]$ (for both point and range plaintext queries), where CT_l and CT_u are the lower bound and upper bound ciphertext values, will be sent to the network. Each node will check whether it has any ciphertext value that satisfies the query Q', and return these ciphertext values if there is any. To ensure successful decryption in Phase-3, we require that there are at least two unique ciphertext values to be returned; if there is only one ciphertext CT value that satisfies Q', the next ciphertext value that is greater than CT will also be sent back, even though it may not satisfy Q'.

Phase-3: Query post-processing at user side. After the user receives the returned ciphertext values $CT_1, CT_2, \cdots, CT_t$ from the network, he/she will decrypt these values and obtain the plaintext values. In particular, with the knowledge of the gap values δ, he/she calculates $s_i = CT_{i+1} - CT_i$, the distance of every two successive ciphertext values (we assume $CT_1 \leq \cdots \leq CT_t$). If there exists any s_i that equals to $w_2 * \delta$, then the user deciphers CT_i as $(CT_i - w_1 * \delta)$. The reason that only w_2 is used for decryption is that if there exists any answer PT, it must satisfy that for the first and the second divided values CT_1 and CT_2 of PT, $CT_1 = PT + w_1 * \delta$, and $CT_2 = PT + (w_1 + w_2) * \delta$, thus there must exist $s_i = CT_{i+1} - CT_i$ that equals to $w_2 * \delta$. If there is no such s_i that equals to $w_2 * \delta$, then there is no answer to the queries. The success of the Phase-3 decryption is guaranteed by: (1) our design of the dividing scheme that requires that w_2 is unique among all weight values, so that is $s_i = w_2 * \delta$, and (2) our Phase-2 query evaluation procedure that requires that at least two ciphertext values (i.e., at least the first and the second divided values) should be returned.

We illustrate the query post-processing procedure through the following example: Given the plaintext values {10,11,13,14,17} with $\delta = 0.5$, and the weights $\{0.1, 0{,}3, 0.2, 0.1\}$, the divided ciphertext values will be $CT = \{10.05, 10.15, 11.05,$ $11.2, 11.3, 13.05, 14.05, 14.2, 14.3, 14.35, 17.05\}$. Let's consider a plaintext query $Q : V \in [13.45, 14.15]$. It is translated to the ciphertext query $Q' : V \in [13.5, 14.5]$.

Applying Q' on CT will return $\{13.05,14.05,14.2,14.3,14.35\}$. There exists two ciphertext values 14.05 and 14.2 whose distance equals to $\delta * w_2 = 0.15$. Thus the value 14.05 is deciphered as $14.05 - w_1 * \delta = 14.05 - 0.1 * 0.5 = 14$.

4 Simulation Evaluation

4.1 Metrics

To evaluate the performance of our proposed 1-to-n encryption strategy for coping with frequency-based attacks, we developed the following metrics.

Overhead by dividing. We would like to measure the additional number of ciphertext values that are introduced during the dividing process. For the case of global frequency-based attack, the overhead by dividing metric is defined as $\frac{m-k}{k}$, where m and k are the total number of distinct ciphertext and plaintext values in the network. For the case of local frequency-based attack, the overhead by dividing metric is defined as $\frac{\sum_{i=1}^{N} m_i - \sum_{i=1}^{N} k_i}{\sum_{i=1}^{N} k_i}$, where m_i and k_i are the number of distinct ciphertext and plaintext values on node $i (1 \leq i \leq N)$. Intuitively, the more the additional number of ciphertext is introduced during dividing, the more computational overhead is incurred. We will evaluate the overhead by dividing under various decipher probability for both global and local frequency-based attacks.

Overhead by emulating. In addition to evaluating the computational overhead introduced by dividing, we are interested in quantifying the additional noise amount for performing emulating in order to achieve uniform distribution of ciphertext values. We define the overhead by emulating as $\frac{S_e - S_o}{S_o}$, where S_o and S_e are the sizes of the data memory before and after emulating.

4.2 Methodology

We conducted simulation of a wireless network with multiple nodes using Matlab. Each wireless node collects the data and stores it on itself. We tested on three network sizes with number of nodes set to $N = 20, 60$ and 100 respectively. For each simulation setup, we controlled the total number of distinct plaintext values in the network to be less than or equal to 200. The occurrence frequencies of plaintext values on each wireless node are positive integer in the range of $[0, 100]$ that follows a uniform distribution. Our simulation results are the average over 100 runs for each simulation setup.

4.3 Coping with Global Frequency-Based Attacks

When coping with global frequency-based attacks, we first study the effectiveness of our scheme under various decipher probability. Figure 2 (a) and (b) present the overhead by both dividing and emulating in the network under various decipher probability when fixing the distinct plaintext values at 200. The key observation in Figure 2 (a) is that the overhead by dividing is always small (under 10%) even when the decipher probability goes to around 10^{-5}. This is encouraging as

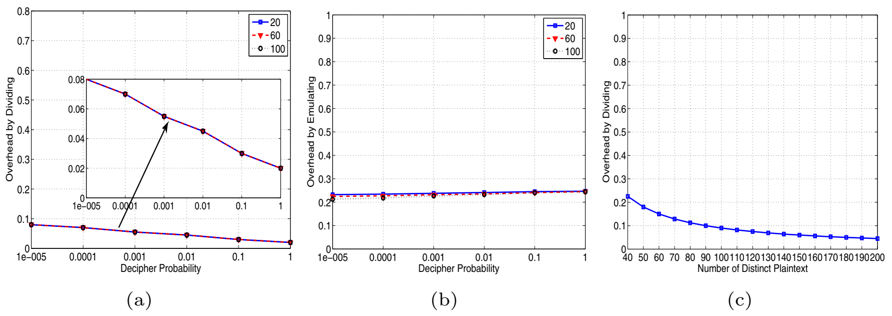


Fig. 2. Effectiveness evaluation when coping with global frequency-based attacks: (a) and (b) overhead by dividing and emulating under various decipher probability when fixing the distinct plaintext values at 200; (c) overhead by dividing under various distinct plaintext values when fixing the decipher probability to 0.01 and $N = 60$

it indicates that our scheme can achieve a robust security guarantee under global frequency-based attacks with little overhead incurred by the dividing technique. The other observation is that the curves for different network sizes overlap. This is because the number of ciphertext values required for dividing does not depend on the number of nodes in the network.

Additionally, we observed that the overhead by emulating is not sensitive to the decipher probability. It goes up slightly from 22% to 25% as the decipher probability increases, which is not significant. We further found that the overhead by emulating does not change with the network size. These discoveries suggest that our scheme is robust in terms of the overhead noise produced by the emulating process when achieving a high security guarantee under various network sizes.

We further investigated the overhead by dividing when varying the number of distinct plaintext values in the network. Figure 2 (c) depicts the overhead by dividing as a function of the number of distinct plaintext values when the decipher probability $P = 0.01$ and the network size $N = 60$. We found that the overhead by dividing is sensitive to the number of distinct plaintext values in the network. Particularly, the overhead by dividing decreases from 22% to 8% when the number of distinct plaintext values increases from 40 to 200 in the network. This indicates that the more distinct plaintext values exist in the network, the less additional computational cost is incurred when using our encryption scheme.

4.4 Coping with Local Frequency-Based Attacks

Next, we turn to examine the performance of our scheme under local frequency-based attacks. Figure 3 presents the total overhead introduced in the network by applying our decryption scheme under local frequency-based attacks when fixing the total number of distinct plaintext values at 200 (however, the number of distinct plaintext values on each node is less or equal to 200). As shown in Figure 3 (a), we observed that the overhead by dividing is comparable to that under global

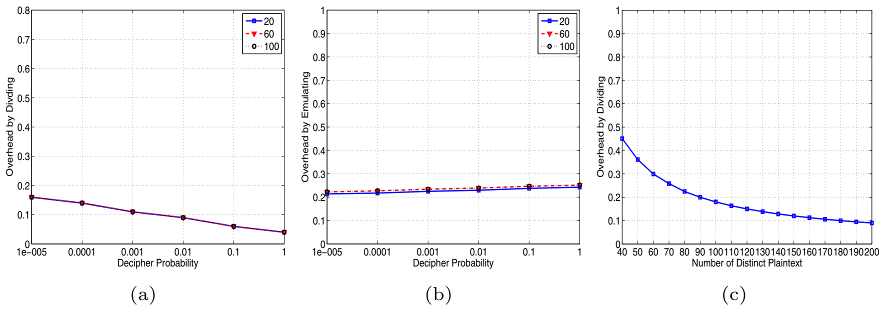


Fig. 3. Effectiveness evaluation when coping with local frequency-based attacks: (a) and (b) overhead by dividing and emulating under various decipher probability when fixing the distinct plaintext values at 200; (c) overhead by dividing under various distinct plaintext values when fixing the decipher probability to 0.01 and $N = 60$

frequency-based attacks. In particular, the overhead by dividing decreases, from 16% to 4%, as the decipher probability increases in a large range from 10^{-5} to 1. Furthermore, the overhead introduced by emulating when varying the decipher probability is presented in Figure 3 (b). We found that the overhead by emulating is not sensitive to the changes of decipher probability and having a slightly increasing trend from 22% to 25% when the decipher probability increases (from 10^{-5} to 1).

These observations are inline with those found in global frequency-based attacks, and indicating that our scheme does not require more overhead when coping with local frequency-based attacks than that under global frequency-based attacks. Additionally, we observed that the overhead do not increase as the network size increases, suggesting that both the additional computational cost and memory overhead introduced by our scheme are stable and will not vary with the network sizes.

Finally, we look at the overhead by dividing as a function of the number of distinct plaintext values in Figure 3 (c) when $P = 0.01$ and $N = 60$. The observation of the declining trend from 45% to 9% as the number of distinct plaintext values increases from 40 to 200 in the network indicates that under local frequency-based attacks, our scheme is sensitive to the number of distinct plaintext values in the network. Furthermore, the overhead by dividing is larger than the corresponding ones under global frequency-based attacks. This is based on our observation that to achieve a given decipher probability, smaller number of distinct plaintext values will require relatively more distinct ciphertext values. Since the uniform distribution of ciphertext values is achieved on each individual node when coping with local frequency-based attacks as opposed to that achieved in the network level under global frequency-based attacks. The overhead ratio by dividing of local frequency-based attacks is higher than that of global frequency-based attacks.

5 Conclusion

In this paper, we proposed a secure data storage scheme that can effectively defend against frequency-based attacks in wireless networks. We considered a sophisticated attack model that the attackers possess the knowledge of frequencies of the original data in the network and utilize such knowledge to decipher the encryption on these data. To cope with such frequency-based attacks, we designed a novel 1-to-n encryption scheme that utilizes our proposed dividing and emulating techniques. We showed that our dividing and emulating techniques not only provide robust security guarantee against frequency-based attacks but also support efficient query evaluation over encrypted data. Our extensive simulation results confirmed the effectiveness and efficiency of our approach.

References

1. Pietro, R.D., Mancini, L.V., Soriente, C., Spognardi, A., Tsudik, G.: Catch me (if you can): Data survival in unattended sensor networks. In: Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom) (2008)
2. Girao, J., Westhoff, D., Mykletun, E., Araki, T.: Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks. Ad Hoc Networks 5, 1073–1089 (2007)
3. Shenker, S., Ratnasamy, S., Karp, B., Govindan, R., Estrin, D.: Data-centric storage in sensornets. ACM SIGCOMM Computer Communication Review archive 33 (2003)
4. Ghose, A., Grossklags, J., Chuang, J.: Resilient data-centric storage in wireless adhoc sensor networks. In: Chen, M.-S., Chrysanthis, P.K., Sloman, M., Zaslavsky, A. (eds.) MDM 2003. LNCS, vol. 2574, pp. 45–62. Springer, Heidelberg (2003)
5. Shao, M., Zhu, S., Zhang, W., Cao, G.: pdcs: Security and privacy support for datacentric sensor networks. In: Proceedings of the IEEE International Conference on Computer Communications (INFOCOM) (2007)
6. Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, J.: Spins: security protocols for sensor netowrks. In: 7th ACM International Conference on Mobile Computing and Networking (2001)
7. Liu, D., Ning, P.: Establishing pairwise keys in distributed sensor networks. In: 10th ACM Conference on Computer and Communications Security (2003)
8. Capkun, S., Hubaux, J.P.: Secure positioning of wireless devices with application to sensor networks. In: Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), pp. 1917–1928 (2005)
9. Chen, Y., Trappe, W., Martin, R.P.: Detecting and localizing wirelss spoofing attacks. In: Proceedings of the Fourth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON) (May 2007)

10. Yang, J., Chen, Y., Trappe, W.: Detecting sybil attacks in wireless and sensor networks using cluster analysis. In: The Fourth IEEE International Workshop on Wireless and Sensor Networks Security (IEEE WSNS) (2008)
11. Joshi, D., Namuduri, K., Pendse, R.: Secure, redundant and fully distributed key management scheme for mobile ad hoc networks: an analysis. EURASIP Journal Wireless Communnication Networks (4), 579–589 (2005)
12. Nalin, S., Yang, C., Zhang, W.: Securing distributed data storage and retrieval in sensor networks. In: 5th Pervasive Computing and Communications (2007)
13. Wang, Q., Ren, K., Lou, W., Zhang, Y.: Dependable and secure sensor data storage with dynamic integrity assurance. In: 28th IEEE International Conference on Computer Communications (2009)
14. Wang, H., Lakshmanan, L.V.: Efficient secure query evaluation over encrypted xml database. In: 32nd International Conference on Very Large Data Bases (2006)
15. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 224–241. Springer, Heidelberg (2010)