Real-time Detection of Clone Attacks in Wireless Sensor Networks

Kai Xing
Department of Computer Science
The George Washington University
Washington, DC 20052, USA
kaix@gwu.edu

Fang Liu
Department of Computer Science
University of Texas - Pan American
Edinburg, Texas 78539 USA
fliu@cs.panam.edu

Xiuzhen Cheng
Department of Computer Science
The George Washington University
Washington, DC 20052, USA
cheng@gwu.edu

David H.C. Du
Department of Computer Science & Engineering
University of Minnesota/NSF
Minneapolis, Minnesota, USA
du@cs.umn.edu

Abstract

A central problem in sensor network security is that sensors are susceptible to physical capture attacks. Once a sensor is compromised, the adversary can easily launch clone attacks by replicating the compromised node, distributing the clones throughout the network, and starting a variety of insider attacks. Previous works against clone attacks suffer from either a high communication/storage overhead or a poor detection accuracy. In this paper, we propose a novel scheme for detecting clone attacks in sensor networks, which computes for each sensor a social fingerprint by extracting the neighborhood characteristics, and verifies the legitimacy of the originator for each message by checking the enclosed fingerprint. The fingerprint generation is based on the superimposed s-disjunct code, which incurs a very light communication and computation overhead. The fingerprint verification is conducted at both the base station and the neighboring sensors, which ensures a high detection probability. The security and performance analysis indicate that our algorithm can identify clone attacks with a high detection probability at the cost of a low computation/communication/storage overhead. To our best knowledge, our scheme is the first to provide realtime detection of clone attacks in an effective and efficient way.

1 Introduction

In sensor networks, adversaries may easily capture and compromise sensors and deploy unlimited number of clones of the compromised nodes. Since these clones have legitimate access to the network (legitimate IDs, keys, other security credentials, etc.), they can participate in the network operations in the same way as a legitimate node, and thus launch a large variety of insider attacks [1, 8, 15], or even take over the network. If these clones are left undetected, the network is unshielded to attackers and thus extremely vulnerable. Therefore, clone attackers are severely destructive, and effective and efficient solutions for clone attack detection are needed to limit their damage.

Nevertheless, detecting clone attacks is not trivial at all. The fundamental challenge comes from the fact that the replicas own all the security information (ID, keys, codes, etc.) of the original compromised sensor. Thus, they can pass all the identity/security check and escape from being distinguished from a legitimate sensor. In addition, a "smart" clone may try to hide from being detected by all means. Furthermore, clones may collude to cheat the network administrator into believing that they are legitimate. Note that an adversary may distribute clone nodes anywhere in the network. Thus localized detection schemes do not work effectively.

Most existing research efforts in sensor networks against clone attacks focus on preventive technologies rather than detective techniques, e.g., key schemes to prevent sensors from being compromised. Unfortunately, most of these preventive technologies (i.e., key schemes) may easily lose their power against clone attacks [3]. Therefore it is imperative to provide effective/efficient clone attack detection. Actually clone detection in sensor networks is a relatively overlooked area. To our best knowledge, the two works [3, 12] that propose nontrivial schemes for clone attack detection rely on having each sensor send its location claims or neighbor list to witnesses or the BS, which requires a large amount of message transmissions. Due to this high message overhead, these schemes can not detect cloned nodes in realtime, therefore could not completely protect the network from clone attacks. In addition, these two schemes are effective only when the number of clones are limited.

Despite the difficulties, detection of clone attacks can be achieved by exploring the social characteristics of each sensor. Note that once deployed, a sensor resides within a fixed neighborhood. The sensor and its neighborhood form a small "community", or a "social network". A cloned sensor can have the same legitimate credentials (ID, keys, etc) as the original node, but cannot have the same community neighborhood. Thus, each sensor can be distinguishably characterized by its social community network. Note that in a small community, a newcomer can be easily recognized if speaking with a different accent. Similarly, a clone node can be easily identified by its neighbors if carrying a "social signature" belonging to a different community. This observation motivates our research on clone sensor detection.

In this paper, we propose a novel scheme to detect clone attacks, in which a fingerprint for each sensor is computed by incorporating the neighborhood information through a superimposed s-disjunct code, and the existence of a clone attack is identified by checking the validity of the originator's fingerprint for each message. The detection scheme is sketched as follows. Before deployment, each sensor is preloaded with a codeword generated from a superimposed s-disjunct code. Then a node can compute its social fingerprint based on the codewords collected from its neighborhood. Each sensor should also compute the fingerprints for the nodes in its local community, and store them for later verification purpose. Whenever a sensor sends a message to the base station, it should include its fingerprint as well. Then the neighboring sensors can verify the legitimacy of the source node by comparing the enclosed fingerprint with their own record. To further reenforce the security, the base station should also retain a fingerprint for each sensor, and monitor the network globally by checking if any inconsistence exists among reports originated from sensors with the same ID.

Compared with the existent work against clone attacks in sensor networks [12] [3], our algorithm has the following characteristics and advantages:

- Our scheme explores the superimposed s-disjunct code for a timely clone attack detection. A fingerprint can be easily encoded with a very short bit stream, which results in small message overhead.

- Our scheme can identify cloned sensors with a high detection accuracy at the expense of a very low communication/computation/storage overhead.

- Our scheme is robust against collaborative clone attackers, and has no limitation on the number of compromised/cloned sensors, which is a significant improvement compared to the existent works [3, 12].

- Our scheme conducts fingerprint verification locally (via neighboring nodes) and globally (via the basestation) for each message broadcasted by any node, therefore clone attackers can be detected in realtime.

The rest of the paper is organized as follows. In Section 2, the most related work on clone attack detection is briefly summarized. Then we introduce the network and security models in Section 3, and present the preliminaries in Section 4. The novel scheme on detecting clone attacks is presented in Section 5, and the security analysis and performance evaluation are reported in Section 6 and Section 7, respectively. Finally, we conclude this paper in Section 8.

2 Related Work

A straightforward solution to defend against clone attacks is to let the base station collect the neighborhood information (e.g. location, neighbor list, etc.) from each sensor and monitor the network in a centralized way.This approach suffers from high communication overhead by requesting redundant information from the network. Further, a "smart" clone may report the neighborhood of the original node, making the base station fail in identifying the replica.

In [1], Capkun et al. propose for one-hop networks that the base station (BS) can store the unique signal characteristic for each device, and thus device cloning can be detected accordingly. However, in a multi-hop sensor network, it is impractical for BS to track the signal characteristics of sensors multi-hops away. In localized voting/misbehavior detection schemes [4, 7, 8], nodes within a neighborhood agree/vote on the legitimacy of a given node based on their local observations. Nevertheless, these schemes are not capable of detecting clones with normal behavior, and may fail when multiple clones in close proximity collude. Furthermore, localized voting/misbehavior detection schemes inherently lack the ability to detect distributed clones that may appear at any place in the network.

To our best knowledge, the first non-naive detection scheme against distributed clone attacks is to employ witness nodes to undertake the task of clone attak detection [12]. For a sensor u, the neighbors should register u's ID and location at multiple witness nodes. The witness nodes can be either randomly selected throughout the network, or simply picked up along a routing path. Any witness node having received conflicting reports about the same sensor should initiate a revoke message. For a high detection probability, this witness-based scheme exploits flooding for information exchange and thus results in a high communication overhead. Also, the scheme relies on public key cryptography, which is expensive for most mote-like sensors. The storage overhead is also high since each sensor needs to store enough public keys of the others.

Another distributed solution is to detect clones based on set operations. In [3], Choi *et al.* propose to divide a sensor network into exclusive subregions and check if there is any overlapping between them. An non-empty intersection indicates the existence of replicated sensors. The results of the membership checking are united and authenticated along a tree structure, and sent to the base station finally. Despite the fact that the number of messages is reduced to $O(N)$, the length of the messages increases linearly, and the total amount of data to be transferred for membership checking is not reduced at all.

As discussed above, the existent solutions have their limited usage for detecting clone attacks in sensor networks, either due to the limited vision on the whole network by the detectors [1, 7, 8], or because of the high communication overhead for information collection to identify and revoke the cloned nodes [12] [3].

3 Network and Security Models

3.1 Network Model

In this paper, we consider a static homogeneous sensor network, in which a base station (BS) intermittently collects data from multi hops away. There exist N resource-constrained sensors in the network, whose positions can be determined after deployment via a self-positioning mechanism such as those proposed in [2, 9, 13]. Let $\mathcal{N}(u)$ denote an open neighborhood of u that contains n nearest neighbors. Note that $\mathcal{N}(u)$ could be the one-hop neighborhood or any neighboring area containing n closest nodes. We denote $\mathcal{N}(\mathcal{N}(u))$ the cumulative neighborhood of $\mathcal{N}(u)$. For any two arbitrary neighboring nodes u and v, we can infer that $\mathcal{N}(u)$ and $\mathcal{N}(v)$ are different ($\mathcal{N}(u) \neq \mathcal{N}(v)$) since $u \notin \mathcal{N}(u)$ but $u \in \mathcal{N}(v)$ and $v \notin \mathcal{N}(v)$ but $v \in \mathcal{N}(u)$.

3.2 Security Model

In our consideration, sensors are not tamper-resistant. The compromise or capture of a sensor releases all its security information to the attacker. Thereafter, the adversary can start replicating the node, and distribute the clones throughout the network. Note that the cloned nodes own all the legitimate information of the compromised node (e.g. ID, keys, code, etc.). Thus the replicas can easily participate in the network operation in the same way as the legitimate nodes. The cloned nodes are under the control of the adversary, and therefore can launch various internal attacks afterward. For example, a cloned node can easily fabricate a false event report to mislead the decision makers, or keep injecting bogus data to cause network outage.

In accordance with the existent work against clone attacks [12] [3], we assume that the adversary cannot create new IDs for cloned nodes, since otherwise the attackers will have to create the corresponding security information (keys, codes, etc.), which is very difficult and even infeasible in most cases. Thus, the adversary would simply select to extract the cryptographic information of a compromised sensor and load into multiple nodes.

We assume that the base station is sufficiently powerful to defend itself against security threats, while the low-cost sensors can be compromised or physically captured. We also assume that a secure routing protocol is available, such that the message being forwarded can be protected from being altered and can be authenticated (e.g. [6, 11, 16]).

4 Preliminaries

In this section, we introduce the basic knowledge on *superimposed s-disjunct codes*, based on which we can retrieve the social community information and create a fingerprint for each sensor. Such fingerprints can help detect clone attacks, which is detailed in the next section.

Let X be a $M \times N$ binary matrix with $X_{i,j}$ being the element at the i-th row and j-th column. In this paper, we consider a matrix X with a constant column weight ω and a constant row weight λ. Then,

$$\sum_{i=1}^{M} X_{i,j} = \omega, \ \sum_{j=1}^{N} X_{i,j} = \lambda,$$

where $1 \leq i \leq M, 1 \leq j \leq N$. The binary matrix X can be used to define a *binary code*, with each column $X_j = (X_{1,j}, X_{2,j}, ..., X_{M,j})^T$ corresponding to a *codeword.*

Definition 4.1 *Given two binary codewords $y = (y_1, y_2 ..., y_M)^T$ and $z = (z_1, z_2, ..., z_M)^T$, we say that y covers z if the Boolean sum (logic OR operation) of y and z equals y, i.e. $y \bigvee z = y$.*

Definition 4.2 *An $M \times N$ binary matrix X defines a superimposed code of length M, size N, strength s $(1 < s < M)$, and $listsize \leq L - 1$ $(1 \leq L \leq M - s)$, if the Boolean sum of any s-subset of columns of X can cover no more than $L - 1$ columns of X which are not in the s-subset. This code is also called an (s, L, M)-code of size N.*

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1
\end{pmatrix}$$

Figure 1. An example of a superimposed $(3, 1, 13)$-code of size 13

For example, the matrix in Fig. 1 defines a superimposed $(3, 1, 13)$-code of size 13.

Definition 4.3 *A binary matrix X defines an s-disjunct code if and only if the Boolean sum of any s-subset of columns of X does not cover any other column of X that are not in the s-subset.*

According to the s-disjunct characteristic of superimposed s-disjunct codes, we can derive the following important property, which will be employed to compute fingerprints to work against clone attacks in the following section:

Property 4.1 Given a superimposed s-disjunct code X, for any s-subset of columns of X, there exists at least one row in X that intersects all the s columns with a value 0.

Note that how to construct a good superimposed s-disjunct code has been extensively studied in literature ([5, 10, 14]). We assume one is available for us to use in this paper. Also, we employ a superimposed s-disjunct code with constant weights in our detection scheme.

5 Real-time Clone Attack Detection in Sensor Networks

In this section, we present our scheme for detecting clone attacks. The detection scheme consists of two phases: computing a fingerprint for each sensor based on its social network, and then detecting clone attacks afterwards.

5.1 Generation of Fingerprints

Before deployment, a superimposed s-disjunct code X is pre-computed offline. As stated in Section 4, X can be represented by an $M \times N$ binary matrix, where N is the number of sensors to be distributed in the network region. Each sensor is preloaded with a unique codeword, which is also a column of the matrix X.

Right after deployment, sensor u broadcasts a message containing u's codeword to the neighborhood $\mathcal{N}(\mathcal{N}(u))$ and listens for the messages sent in the neighborhood $\mathcal{N}(\mathcal{N}(u))$. In our consideration[1], the neighborhood $\mathcal{N}(u)$ should satisfy $n \geq s$, where n is the number of sensors in $\mathcal{N}(u)$, s is the strength of the superimposed code X.

After the information collection, sensor u computes the fingerprint for each node $v \in \mathcal{N}(u)$, and stores the fingerprint for later use (to be explained in Section 5.2). Given a sensor $v \in \mathcal{N}(u)$, sensor u computes v's fingerprint as follows. Let $X^{(v)} = \{X_1^{(v)}, X_2^{(v)}, ..., X_n^{(v)}\}$ denotes the codeword set of the nodes in v's neighborhood $\mathcal{N}(v)$, where $X_i^{(v)}$ denotes the codeword of v's i-th closest neighbor. Among the received codewords $X^{(v)}$, the Boolean sum of $X_{(s)}^{(v)}$, the codeword set of v's s-closest neighbors, is computed first. According to the property of the superimposed s-disjunct code (see Property 4.1 in Section 4), the resulting vector should contain at least one element with a value '0'. These zero elements imply the relationship among the s neighbors, which can be employed to represent the social characteristic of sensor v. Motivated by this observation, we use the binary representation of the position of one zero element in the Boolean sum of $X_{(s)}^{(v)}$ as the social fingerprint of v. Since the Boolean sum of $X_{(s)}^{(v)}$ may contain multiple zero elements, we propose to employ more codewords in $\mathcal{N}(v)$ for the purpose of decreasing the number of zero elements. Intuitively, the social fingerprint should be "stronger" if more information from $\mathcal{N}(v)$ is brought in during the fingerprint computation. Algorithm 1 details the procedure of fingerprint computation for each sensor by investigating the associated neighborhood information.

Sensor u uses Algorithm 1 to compute the fingerprint for each node $v \in \mathcal{N}(u)$. Algorithm 1 takes the codeword set $X^{(v)}$ as input sensor u has received from its neighborhood. The algorithm starts from an s-subset of $X^{(v)}$ that contains the codewords of the s closest neighbors of sensor v, and expands the subset until any further increment will cause the resulting boolean sum to contain no zero. For the subset resulting from the last increment, compute the boolean sum and select one of the zero elements, the position of which will be the fingerprint of sensor v.

[1]As stated in Algorithm 1, sensor u may require more than s codewords for fingerprint computation. Thus, n should be large enough for sensor u to collect the necessary information from $\mathcal{N}(u)$.

Algorithm 1 Fingerprint Creation

```
1: function $FP$=fingerprint($v, X^{(v)}$)    ▷ Compute the fingerprint for sensor $v$ based on the codeword set $X^{(v)} = \{X_1^{(v)}, ..., X_n^{(v)}\}$. Note that sensor $u$ should conduct this algorithm for each neighbor $v \in \mathcal{N}(u)$.
2:     $B = X_1^{(v)} \bigvee X_2^{(v)} \bigvee \cdots \bigvee X_s^{(v)}$    ▷ Compute the boolean sum of $X_{(s)}^{(v)}$, the codeword set of $v$'s $s$-closest neighbors
3:     $i = s + 1$
4:     while $(B \bigvee X_i^{(v)}) \neq \vec{1}$ do    ▷ Expand $X_{(i)}^{(v)}$ by adding the codeword of $v$'s $i$+1-closest neighbor, until any further increment causes the boolean sum to contain no zeros.
5:         $B = B \bigvee X_i^{(v)}$
6:         $i = i + 1$
7:     end while
8:     $posZeros = findZeros(B)$    ▷ Return the positions of the zero elements in the boolean sum $B$
9:     if $size(posZeros) > 1$ then
10:        $k = pseudoRansom(v, size(posZeros))$
11:        $position = Select(k, posZeros)$    ▷ Select the $k$-th zero element from posZeros if there are more than one zero element in $B$
12:    else
13:        $position = posZeros$
14:    end if
15:    $FP = binary(position)$    ▷ return the binary representation of the position with a zero element
16:    return $FP$
17: end function
```

Note that for v and a neighbor $u \in \mathcal{N}(v)$, u and v both need to conduct Algorithm 1 to compute the fingerprint FP_v for sensor v. By taking v's ID as input for the pseudo random function (see line 10 in Algorithm 1), u and v will reach the same FP_v though they need to select one zero element from multiple options independently.

5.2 Detection of Clone Attacks

For sensor u, its fingerprint is computed from the codewords collected from its neighborhood $\mathcal{N}(u)$. As stated in Section 3, sensors are stationary after deployment. A legitimate sensor u belongs to a "fixed" neighborhood, whose social characteristics can be encoded into u's fingerprint. Therefore, each sensor is required to "sign" with its fingerprint FP_u whenever it generates a new message to send to the base station. The message transmission should be in the following format[2]:

$$u \rightarrow BS : \{ID_u, FP_u, content\}$$

[2]This message will be protected by encryption and message authentication code, as stated in our security model.

Assume X is the superimposed s-disjunct code to generate the social codeword for each sensor, which can be represented by an $M \times N$ matrix. According to Algorithm 1, the length of a fingerprint is $\log_2(M)$. Even with $M = 100,000$, a fingerprint takes no more than 2 bytes to be included in a message. Hence, our detection algorithm imposes a very slight message overhead for protecting a sensor network against clone attacks.

In our consideration, a cloned sensor may use an arbitrary fingerprint (e.g. the fingerprint of the original sensor), or compute a new fingerprint that is consistent with its new residency. Hence, detecting clone attacks should be conducted in two aspects:

5.2.1 Detection at the sensor side

Suppose u generates a new message, which is forwarded to a neighbor $v \in \mathcal{N}(u)$. Sensor v should check whether the enclosed fingerprint FP_u is consistent with the one in its record that v has computed for u previously. Once v identifies any mismatch, v should raise an alarm to the base station. Then, the base station should send a query to the neighborhood $\mathcal{N}(u)$. Each sensor in $\mathcal{N}(u)$ should reply to BS with its own record about FP_u. Thus, BS can determine which sensor should be revoked afterwards[3].

The local fingerprint verification ensures that no sensor can use a fingerprint that is not consistent with its neighborhood. Note that a legitimate sensor u derives its fingerprint based on the information retrieved from its neighborhood. The local information exchange ensures that u's neighbors can also compute u's fingerprint independently. Thereafter, though a cloned sensor u can "pretend" to be legitimate by having all the valid security information, it cannot cheat its neighbors that can easily tell whether u is using an inconsistent fingerprint.

5.2.2 Detection at the base station

The base station should maintain a fingerprint file indexed with sensor IDs, and insert an entry for sensor u upon receiving u's first message. After BS receives a new message C from sensor u, it checks whether the fingerprint FP_u in C matches its record obtained from u's previous messages. If FP_u does not match the record, there must be a clone attack in the network. Then, BS may broadcast a revoke message concerning sensor u throughout the network, such that those sensors with the ID u will be isolated afterwards.

The detection at the base station is to work against a "smart" clone that intelligently computes a new fingerprint consistent with its current neighborhood so as to escape

[3]In this case, either u is a clone, or v is a compromised node which raises a false alarm.

from being identified by the neighboring sensors. By establishing a fingerprint file, the base station can easily determine whether there exist several sensors in the network that use different fingerprints but with the same ID.

6 Security Analysis

In this section, we analyze the impact of compromised and cloned nodes on our detection algorithm. We observe that an adversary can launch effective clone attacks at the following two scenarios. Note that a clone attack at other scenarios can be detected via the fingerprint computed by Algorithm 1 with a much less effort.

- Node compromise/clone during fingerprint generation,
- Node compromise/clone during the detection phase.

Next, we will study the resilience of our scheme under these two scenarios, and quantify the effectiveness of our detection by studying the detection probability.

6.1 Node compromise/clone at fingerprint generation

Assume an adversary compromises a sensor u right after deployment, replicates and distributes the clones before the fingerprint generation is finished in the network. Then u's clones, say $u'_1, u'_2, ..., u'_t$, can participate in the fingerprint generation procedure as a legitimate sensor. Since the clones are deployed into different locations, their derived fingerprints will be different. Thus, the base station can easily identify these clones, which have the same ID ($ID_u = ID_{u'_1} = ... = ID_{u'_t}$) but different fingerprints.

Note that there is no impact on a legitimate sensor w, if a clone node u'_i is inserted into the neighborhood $\mathcal{N}(w)$. Sensor w can safely use its fingerprint which may contain the codeword contributed by the clone u'_i though, since the other legitimate neighbors in $\mathcal{N}(w)$ will use the codeword of u'_i as well. It does not affect the effectiveness of w's fingerprint against clone attacks, because w's fingerprint is based on the neighborhood difference and social relationships rather than an individual codeword.

6.2 Node compromise/clone at the detection phase

Assume an adversary compromises a sensor u, replicates and distributes the clones after all the legitimate sensors have derived their fingerprints. Then for a cloned sensor v, the adversary may determine v's fingerprint FP_v according to the following methods:

- *Case I: Sensor v selects $FP_v = FP_u$.* Suppose sensor v generates a message C and forwards it to a neighbor $w \in \mathcal{N}(v)$. If w is legitimate, w should raise an alarm since no match can be found in w's fingerprint records. Then the base station can identify the clone v after checking the fingerprints belonging to $\mathcal{N}(v)$. Unless the adversary completely compromises and controls the neighborhood $\mathcal{N}(v)$, the clone v will be identified by our detection scheme.

  Note that there is no incentive for the adversary to compromise all nodes in $\mathcal{N}(v)$ in order to launch a clone attack. Furthermore, for a cloned area (containing cloned nodes only) that is larger than a typical open neighborhood, all the boundary nodes can be easily identified and then revoked. Thus, the whole compromised/cloned region will be isolated. Our detection scheme is robust against colluding attackers.

- *Case II: Sensor v selects an arbitrary bit stream as FP_v.* Same as *case I.*

- *Case III: Sensor v computes FP_v based on the codewords from its neighborhood $\mathcal{N}(v)$.* A smart clone tries to escape from being identified by its neighbors, and computes a fingerprint consistent with its neighborhood. Assume the adversary is powerful enough to listen on the codewords broadcasted around $\mathcal{N}(v)$ and help v compute its FP_v. However, after receiving messages from sensors u and v, the base station will find out that $ID_u = ID_v$ but $FP_u \neq FP_v$. Then the base station identifies the existence of a clone attack, and therefore revokes all the sensors with ID_u.

6.3 Detection Probability

In the following, we investigate the probability $P_{undetected}$ that a clone node escapes from being detected successfully. Assume the adversary compromises a sensor u, clones t copies of u (denoted as $u'_1, u'_2, ..., u'_t$), and distributes the clones into the network. To avoid being detected, these t clones must fulfil the following two requirements simultaneously:

- *Condition I*: All the clones $u'_1, u'_2, ..., u'_t$ must use the same fingerprint as the sensor u. Otherwise, the base station will identify the difference among the fingerprints used by these nodes that share the same IDs.

- *Condition II*: Each of the clones $u'_1, u'_2, ..., u'_t$ must use a fingerprint that is consistent with its current neighborhood. Otherwise, the cloned node will be identified by their neighbors.

Thereafter, only when the neighbors of the t clones contribute to the same fingerprint as that of sensor u, our detection algorithm fail to identify.

Let X be the superimposed s-disjunct code with a constant column weight w and a row weight λ. Let $X_{(r)}^{(u)} = \{X_1^{(u)}, ..., X_r^{(u)}\}$ be the codeword set selected for u's fingerprint in Algorithm 1, where $X_i^{(u)}$ (a column of X) is the codeword of u's i-th closest neighbor in $\mathcal{N}(u)$ and $r \geq s$. Denote $B^{(u)}$ as the boolean sum of $X_{(r)}^{(u)}$. We first study $P(B_1^{(u)} = 0|r = i)$, the probability that the 1st element in the vector $B^{(u)}$ is zero given that $X_{(r)}^{(u)}$ contains i columns of X. Then,

$$P(B_1^{(u)} = 0|r = i) \quad = \quad \frac{\binom{(N-\lambda)}{i}}{\binom{N}{i}}. \tag{1}$$

The same results can be obtained for $P(B_2^{(u)} = 0|r = i)$, $\cdots$, $P(B_M^{(u)} = 0|r = i)$. Therefore,

$$\begin{aligned} & P(B_1^{(u)} = 0|r = i) \\ = \; & P(B_2^{(u)} = 0|r = i) \\ = \; & ... \\ = \; & P(B_M^{(u)} = 0|r = i) \end{aligned} \tag{2}$$

It also holds true that,

$$\textstyle\sum_{k=1}^{M} P(B_k^{(u)} = 0|r = i) = 1. \tag{3}$$

According to Eqs. (2) and (3), we have

$$\begin{aligned} & P(B_k^{(u)} = 0|r = i) = 1/M, \\ & \text{where } k = 1, 2, ..., M,\ i \geq s. \end{aligned} \tag{4}$$

Let i (j) denote the number of codewords that sensor u (v) uses for the computation of the fingerprint FP_u (FP_v). Then we study the probability that sensors u and v run Algorithm 1 independently and obtain the same fingerprints $FP_u = FP_v$, which is

$$\begin{aligned} & P(FP_u = FP_v) \\ = \; & \sum_{k=1}^{M} P(B_k^{(u)} = 0|r = i) \cdot P(B_k^{(v)} = 0|r = j) \\ = \; & M \times \frac{1}{M} \times \frac{1}{M} \\ = \; & \frac{1}{M} \end{aligned} \tag{5}$$

Now we study the probability that a compromised node u and its t clones, namely $u'_1, u'_2, \cdots, u'_t$, derive the same fingerprints. Let j_i be the number of codewords that the clone node u'_i uses for its fingerprint generation. Then,

$$\begin{aligned} & P_{undetected} \\ = \; & P(FP_u = FP_{u'_1} = \ldots = FP_{u'_t}) \\ = \; & \sum_{k=1}^{M} P(B_k^{(u)} = 0|r = i) \cdot P(B_k^{(u'_1)} = 0|r = j_1) \\ & \ldots P(B_k^{(u'_t)} = 0|r = j_t) \\ = \; & M \times \frac{1}{M} \times \frac{1}{M} \times \ldots \times \frac{1}{M} \\ = \; & \frac{1}{M^t} \end{aligned} \tag{6}$$

Therefore, given a compromised node u and all its t clones, with our algorithm, the detection probability $P_{detected}$ against these clones is

$$\begin{aligned} P_{detected} &= 1 - P_{undetected} \\ &= 1 - \frac{1}{M^t} \end{aligned} \tag{7}$$

Note that M represents the number of rows in the superimposed s-disjunct code X. According to superimposed code construction methods [10] [5], we can generate an $M \times N$ superimposed s-disjunct code with M as large as N, where N is the number of sensors in the network. For a typical sensor network, N is a very large number. Therefore, $P_{detected}$ should be very close to 1. Our detection algorithm can protect sensor network against clone attacks effectively with a high detection probability.

7 Performance Analysis

7.1 Overhead

In the initialization step of our scheme, each node needs to collect the codewords from its local neighborhood, and computes its fingerprint. Therefore, the fingerprint generation poses $O(N)$ local message transmissions in the network overall[4]. Let num_m denote the total number of regular data messages generated in the network during network lifetime, the total message transmission cost in the entire network is $O(num_m \cdot \sqrt{N})$.

Unlike other detection schemes against distributed clone attack (Section 2) that have each node send a separate message rather than regular data message, our scheme attaches each regular message with a corresponding fingerprint. According to Algorithm 1, the fingerprint is pretty tiny. Its size can be bounded by $\log_2 M$, where M is the number of rows in the superimposed s-disjunct code used in the network. Let $ratio$ denote the ratio of the fingerprint size versus the regular packet size, and L_{packet} denote the bit-length of a regular message. Then, the message overhead of our scheme in the entire network after

[4]The initialization cost of each scheme in Table 1 is neglected, since the initialization of all the schemes in Table 1 is done in a localized manner, and incurs negligible cost.

Schemes	Message Cost (in bits)	Storage Cost
Broadcast [12]	$C + O(N^2) \cdot T \cdot L_{packet}$	$O(d)$
Deterministic Multicast [12]	$C + O(\frac{glng\sqrt{N}}{d}) \cdot T \cdot L_{packet}$	$O(g)$
Randomized Multicast [12]	$C + O(N^2) \cdot T \cdot L_{packet}$	$O(\sqrt{N})$
Line-Selected Multicast [12]	$C + O(N\sqrt{N}) \cdot T \cdot L_{packet}$	$O(\sqrt{N})$
SET [3]	$C + O(N) \cdot T \cdot L^*_{nonconstant}$	$O(d)$
Our Scheme	$C \cdot (1 + ratio)$	$O(d) + \min(M, \omega \cdot \log_2 M)$

Table 1. Overhead of clone attack detection approaches.

initialization is $O(num_m \cdot \sqrt{N}) \cdot L_{packet} \cdot ratio$, where $ratio = \frac{\log_2 M}{L_{packet}} \times 100\%$.

Let d denote the average size of the neighbor set in the network. Since each node u stores the fingerprints of its neighbors, the average memory cost is $O(d) + \max(M, \omega \cdot \log_2 M)$, where $\max(M, \omega \cdot \log_2 M)$ denotes the memory usage for u to store its social codeword, ω represents the column weight in the superimposed s-disjunct code.

Note that only simple binary operations are involved in local fingerprint computation, therefore our scheme has extremely low computation overhead.

7.2 Comparison with Existent Work

Let $C = O(num_m \cdot \sqrt{N}) \cdot L_{packet}$ denote the total message transmission cost (in bits), and T denote the number of runs of other clone attack detection schemes during sensor network lifetime, as shown in Table 1. Compared with these schemes, our scheme incurs a low communication overhead and a comparative storage overhead, which are $C \cdot ratio$ and $O(d) + \max(M, \omega \cdot \log_2 M)$, respectively.

In particular, compared with SET, despite that the number of message transmissions is reduced to $O(N)$ in SET, the message length increases linearly (denoted by $L^*_{nonconstant}$), and the total amount of information (membership) transmitted in the entire network is at the same order of $O(N) \cdot T \cdot L^*_{nonconstant} = O(N\sqrt{N}) \cdot T \cdot L_{packet}$.

8 Conclusion

In this paper, we present a novel realtime detection scheme against clone attacks. Our algorithm is superior in that a high detection accuracy as well as resiliency can be achieved at the cost of a low communication/computation/storage overhead. In addition, realtime clone detection is conducted whenever messages flow in the network, and cloned attackers can be identified in a very efficient and effective way. To our best knowledge, our scheme is the first to provide realtime detection against clone attacks. For future work, we are going to explore other kinds of social fingerprints and extend our scheme to other classes of networks.

References

[1] S. Capkun and J.-P. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *IEEE Infocom'05*, 2005.

[2] X. Cheng, A. Thaeler, G. Xue, and D. Chen. Tps: A time-based positioning scheme for outdoor sensor networks. In *IEEE INFOCOM*, HongKong China, 2004.

[3] H. Choi, S. Zhu, and T. Laporta. Set: Detecting node clones in sensor networks. In *SecureComm'07*, 2007.

[4] M. Ding, D. Chen, K. Xing, and X. Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *IEEE INFOCOM*, Miami, Florida, 2005.

[5] A. G. D'yachkov and V. V. Rykov. Optimal superimposed codes and designs for renyi's search model. *Journal of Statistical Planning and Inference*, 100(2):281–302, 2002.

[6] M. Kefayati, H. R. Rabiee, S. G. Miremadi, and A. Khonsari. Misbehavior resilient multi-path data transmission in mobile ad-hoc networks. In *SASN '06*, pages 91–100, 2006.

[7] P. Kyasanur and N. H. Vaidya. Detection and handling of mac layer misbehavior in wireless networks. In *IEEE DSN*, 2002.

[8] F. Liu, X. Cheng, and D. Chen. Insider attacker detection in wireless sensor networks. In *IEEE INFOCOM'07*, Anchorage, Alaska, 2007.

[9] F. Liu, X. Cheng, D. Hua, , and D. Chen. Tpss: A time-based positioning scheme for sensor networks with short range beacons. In *ICCNMC'05*, ZhangJiaJie, Hunan, China, 2005.

[10] A. J. Macula. A simple construction of d-disjunct matrices with certain constant weights. *Discrete Math.*, 162(1-3):311–312, 1996.

[11] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *MobiCom '00*, pages 255–265, 2000.

[12] B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In *SP '05*, pages 49–63, 2005.

[13] A. Thaeler, M. Ding, and X. Cheng. itps: An improved location discovery scheme for sensor networks with long range beacons. In *Special Issue of* JPDC, 2004.

[14] K. Xing, X. Cheng, L. Ma, and Q. Liang. Superimposed code based channel assignment in multi-radio multi-channel wireless mesh networks. In *MobiCom '07*, pages 15–26, 2007.

[15] Y. Yang, X. Wang, S. Zhu, and G. Cao. Sdap: a secure hop-by-hop data aggregation protocol for sensor networks. In *MobiHoc '06*, pages 356–367, 2006.

[16] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route detection and filtering of injected false data in sensor networks. In *IEEE INFOCOM*, 2004.