Traffic Decorrelation Techniques for Countering Colluding Eavesdroppers in WSNs

Alejandro Proaño, Loukas Lazos, and Marwan Krunz
Dept. of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, USA
E-mail:{aaproano, llazos, krunz}@ece.arizona.edu

APPENDIX

A. Proof of Proposition 1

Proposition 1: When sensors transmit one packet at a random time within an interval of \( T \), the average number of hops that a packet can traverse per \( T \) is 1.72.

Proof: Consider a source \( v \) with a hop-count \( \eta = |p(v, u)| \) to the destination \( u \). Nodes in \( p(v, u) \) randomly select their transmission times within an interval \( T \). Let the randomly selected transmission times be denoted by \( t_1, t_2, \ldots, t_\eta \). We compute the average number of hops \( (H) \) traversed over \( T \) by a packet \( m \) originating at \( v_1 \), using a combinatorial approach. The value of \( H \) depends on the arrangement order of \( t_1, t_2, \ldots, t_\eta \). All possible arrangements \( n_\eta ! \) occur with equal probability, as the transmission times of each sensor are randomly and independently selected within \( T \). A packet traverses \( \eta \) hops during \( T \), only if \( t_1 < t_2 < \ldots < t_\eta \). This event occurs with probability,

\[
\Pr(H = \eta) = \frac{1}{\eta!}.
\]

Similarly, to find the probability of traversing \( \eta - 1 \) hops, we consider the number of arrangements that satisfy \( t_1 < t_2 < \ldots < t_{\eta-1} \), but not \( t_{\eta-1} < t_\eta \). Thus, we obtain,

\[
\Pr(H = \eta - 1) = \frac{1}{\eta!} \left( \frac{\eta!}{(\eta - 1)!} - 1 \right).
\]

For an arbitrary number of hops \( x \leq \eta - 1 \) we get,

\[
\Pr(H = x) = \frac{1}{\eta!} \left( \frac{\eta!}{x!} - \frac{\eta!}{(x+1)!} \right) = \frac{x}{(x+1)!}.
\]

Computing the expectation of \( H \)

\[
E(H) = \frac{\eta}{\eta!} + \sum_{x=1}^{\eta-1} \frac{x^2}{(x+1)!} + \frac{1}{(\eta-1)!} + \sum_{x=1}^{\eta-1} \frac{x^2}{(x+1)!}.
\]

By applying the ratio test of series on the second term of \( E(H) \), we obtain

\[
\lim_{x \to \infty} \frac{(x+1)^2}{(x+2)!} = \lim_{x \to \infty} \frac{(x+1)^2 + 1}{(x+2)!} = \lim_{x \to \infty} \frac{x^2 + 2x + 1}{x^3 + 2} \leq 1,
\]

which proves that \( E(H) \) is a converging series. Finally, we compute the convergence value of \( E(H) \) via numerical analysis for all \( \eta \geq 1 \),

\[
E(H) = 1.72.
\]

\( \square \)

B. Proof of Proposition 2

Proposition 2: Stage 3 terminates in less than \( \delta_{\text{max}} + 1 \) iterations, where \( \delta_{\text{max}} \) is the maximum degree of any node in the network.

Proof: By contradiction. Assume that Algorithm 2 does not terminate. That is, there exist a \( v \in V \) that is gray for each iteration of Algorithm 2. Based on Step 3 of Stage 1, a node \( u \) becomes black if

\[
v = \arg \max_{u \in \mathcal{N}_v} \left\{ \frac{\delta_u^x(u)}{\delta_{\text{max}}^x(u)} \times \frac{1}{f(u) + 1} \right\}.
\]

In the worst-case scenario, \( v \) is a leaf node that cannot become black during Stage 2. Each time Stage 1 is executed, \( v \) becomes gray and is dominated by a node \( u \in \mathcal{N}_v \) if,

\[
\delta_u^x(v) \leq \frac{\delta_u^x(u)}{\delta_{\text{max}}^x(u)} \times \frac{1}{f(u) + 1},
\]

or,

\[
\delta_u^x(v) \leq \frac{\delta_u^x(u)}{\delta_{\text{max}}^x(u)} \times \frac{\delta_{\text{max}}^x(v)}{f(u) + 1},
\]

where \( \delta_{\text{max}}^x(v) \) is the maximum degree in \( \mathcal{N}_v \) and \( \delta_{\text{max}}^x(u) \) is the maximum degree in \( \mathcal{N}_u \). Note that because \( G \) is connected, \( \delta^x(u) \geq 1 \), and

\[
1 \leq \frac{\delta_u^x(u)}{\delta_{\text{max}}^x(u)}.
\]
Hence,
\[ 1 \leq \frac{\delta^*(v)}{f(u)+1}. \]

Because we assumed Algorithm 2 does not terminate and \( \delta^*(v) \) is finite, at some iteration \( n' \) of Stage 3, \( f(u) = \delta^*(v) \) for all \( u \in N_v \). Therefore, during iteration \( n' + 1 \) we have \( \frac{\delta^*(v)}{f(u)+1} < 1 \) and,
\[ \delta^*(v) \geq \frac{\delta^*(u)}{\delta^*(v)} \times \frac{\delta^*(v)}{f(u)+1}. \]

In this case, \( v \) becomes black during Stage 1, contradicting the assumption that \( v \) is always gray. Moreover, \( n' = \delta^*(v) \times \delta^*(v) \). Also, note that in the worst case \( \delta^*(v) = 1 \) and \( \delta^*(v) = \delta_{\text{max}}, \) where \( \delta_{\text{max}} \) is the maximum degree in the network. Thus, \( n' = \delta_{\text{max}} \) and the maximum number of iterations of Stage 3 is \( \delta_{\text{max}} + 1 \).

**C. Proof of Proposition 3**

**Proposition 3:** The problem of finding an SS-MCDS in arbitrary graphs is NP-complete.

**Proof:** We first show that the problem of computing a single SS-MCDS is NP-complete. To do so, we prove that SS-MCDS is both in NP and it is at least as hard as the Minimum Shortest-path Steiner arborescence (MSPSA) problem, that is known to be NP-complete [1]. We first define the MSPSA.

**Definition 1 (Minimum Shortest-path Steiner arborescence):**

Given a graph \( G(V, E(V)) \) with positive edge weights, a target set \( T \subseteq V \) and a unique root node \( \mu \in T \), a shortest path Steiner arborescence \( S \) is a Steiner tree rooted at \( \mu \), spanning all vertices in \( T \) such that each path \( p(u, \mu) \) is a shortest path in \( G(V, E(S)) \), for all \( u \in T \). The arborescence of smallest cardinality is called minimum shortest-path Steiner arborescence (MSPSA) [2].

The following verifier for the SS-MCDS problem runs in polynomial time in the size of the input SS-MCDS (\( D \)). In the verifier, \( p(v, \mu) \) denotes the shortest path between \( u \) and the root node \( \mu \) in \( G(V, E(V)) \), while \( p_D(u, \mu) \) denotes the shortest path between \( u \) and \( \mu \) in \( G(D, E(D)) \).

**SS-MCDS Verifier:**

**Input:** \( G(V, E(V)), D, \) and \( p(v, \mu) \) for all \( v \in V \).

**Execution:** If the following are true accept, else reject:
1. For all \( v \in V \), either \( v \) is in \( D \), or there exist a \( u \in D \) for which \( u \in N(v) \)
2. Nodes in \( D \) form a connected subgraph, if so obtain \( p_D(u, \mu) \) for all \( u \in D \)
3. For all \( u \in D \), \( |p(u, \mu)| = |p_D(u, \mu)| \)

In Step 1, the SS-MCDS verifier checks whether \( D \) satisfies the DS property. If \( v \) belongs to \( D \), the check requires only one computation. However, in the worst case, \( v \) is not in \( D \) and the algorithm goes through \( v \)'s 1-hop neighborhood looking for a \( u \in D \). Since the maximum size of \( N(v) \) is \( |V| - 1 \), the maximum cost of this operation is \( O(|V|^2) \), which makes \( T = \{ \mu, v_7 \} \). To generate \( G'(\nu', E(\nu')) \), we first define the set of virtual nodes \( \nu = \{ \nu', v_7 \} \) and their respective 0-cost links \( E_1 = \{ (\mu, \nu), (v_7, \nu) \} \). Finally, we add links \( E_2 = \{ (v_1, v_2), (v_4, v_5), (v_3, v_6) \} \) of unitary cost, to connect nodes with no links in \( E(\nu) \) that have the same hopcount to \( \mu \). The resulting graph \( G'(\nu', E(\nu')) \) is shown in Fig. 1(b).
We first prove that if \((D, K)\) is an instance of the SS-MCDS in \(G'\), where \(K = |D|\), it is also an instance of the MSPSA in \(G\).

1) Each vertex \(v' \in \mathcal{X}\) is dominated by its respective vertex \(v \in \mathcal{T}\).

2) The shortest path property of the SS-MCDS guarantees that nodes in \(T\) have a shortest path with respect to \(\mu \in D\). Therefore, it follows that \((D, K)\) is an instance of the MSPSA problem.

Conversely, if \((D, K)\) is an instance of the MSPSA problem in \(G\), we show that \((D, K)\) is an instance of the SS-MCDS problem in \(G'\). First, note that since \(\mathcal{E} \subset \mathcal{E}'\), set \(D\) also induces a tree rooted at \(\mu\) in \(G'\) that includes the shortest path between \(\mu\) and vertices in \(T\). The rest of the proof is as follows,

1) As \(D\) is rooted at \(\mu\), it follows that the shortest path property (with respect to \(\mu\)) is satisfied for all vertices in \(D\).

2) By the Steiner arborescence definition \(\mathcal{T} \subseteq D\). Thus, vertices in \(\mathcal{X}\) are dominated by their respective vertices in \(\mathcal{T}\).

3) Note that set \(\mathcal{T}\) includes the node with the highest hop count \(h^\text{max}\) to \(\mu\). It follows that \(D\) contains vertices with hop counts of \(1, 2, \ldots, h^\text{max}\), due to the shortest path property. Since edges in \(\mathcal{E}_2\) connect all vertices with the same distance to \(\mu\), all vertices in \(\mathcal{V}'\setminus \mathcal{X}\) are dominated as well.

4) Therefore, \(D\) of size \(K\) is a CDS that contains the shortest paths of all vertices in \(D\) with respect to \(\mu\). This makes \((D, K)\) an instance of the SS-MCDS problem, which concludes both proof directions.

\[\square\]

D. Proof of Proposition 4

**Proposition 4:** The message complexity for partitioning the WSN to SS-MCDSs using Algorithm 5 is \(O(\delta^3_\text{max}|V|)\).

Partitioning the WSN to SS-MCDSs (Algorithm 6) yields the same complexity.

**Proof:** Algorithm 5 is executed in three stages: the DS generation stage, the MCDS Approximation stage, and the MCDS update stage. The overhead of each stage is analyzed as follows.

**Stage 1 – DS generation:** In Stage 1, a node \(v \in V\) broadcasts packets in three occasions: (a) in Step 1 when all nodes share their initial values of \(m(v), \delta^*,\) and \(r(v)\), (b) in Steps 3 or 5, when a node changes colors from white to black or gray, and (c) in Step 4 when a white neighbor of \(v\) has become gray. Cases (a) and (b) occur only once per node, yielding the transmission of 2 packets. Case (c) can occur up to \(\delta(v)\) times, as \(v\) may become black after all its neighbors have become gray. Thus, during Stage 1, a node \(v\) may transmit up to \(\delta(v) + 2\) packets. For a maximum node degree \(\delta_{\text{max}}\), this stage has a message complexity of \(O(\delta_{\text{max}}|V|)\).

**Stage 2 – MCDS approximation:** In Stage 2, a node \(v \in V\) transmits in the following occasions: (a) in Step 1 when nodes broadcast the value of \(b\), (b) in Step 2, when a gray node becomes black, (c) in Step 3 when a black node is dominated by a newly-converted black node, and (d) in Step 4 when a gray node overhearing the change of color from gray to black. Cases (a), (b), and (c) can occur only once per node. However, (d) can occur multiple times as all black nodes in the neighborhood of \(u\) can be dominated by other nodes, making \(v\) transmit \(b(v)\) packets. Thus, the total number of packets that a node \(v\) can transmit during Stage 2 is \(b(v) + 3\). Moreover, all transmissions must be relayed to the two-hop neighborhood of \(v\). Therefore, the total number of transmissions generated in Stage 2 is,

\[\delta(v)(b(v) + 3)|V| \leq (\delta(v)^2 + 3\delta(v))|V| \leq (\delta_{\text{max}}^2 + 3\delta_{\text{max}})|V|\]

This stage has a message complexity of \(O(\delta_{\text{max}}^2|V|)\).

**Stage 3 – MCDS update:** In Stage 3, stages 2 and 3 are repeated until all nodes belong to at least one MCDS.

---

**Fig. 2:** Transmission schedule for a path of length \(4h - 3\) according to the DFAS algorithm.
According to Proposition 2 of the manuscript, it may take up to \( \delta_{\text{max}} \) iterations for Stage 3 to terminate. Therefore, the combined overhead of stages 1 and 2 is multiplied by \( \delta_{\text{max}} \).

Combining the overhead of all three stages, we obtain the message complexity for Algorithm 5 as \( O(3^\delta_{\text{max}}|V|) \).

Algorithm 6 has the same complexity as Algorithm 5 because the same stages are executed. The two algorithms differ only in the CDS node selection criteria of Stage 2. In case of Algorithm 5, the gray node that interconnects the highest number of black nodes is changed to black. In case of Algorithm 6, the selection is restricted to nodes that induce shortest paths. Because nodes can change color only once (except in the pruning stage), the two steps in the respective algorithms have the same message overhead. Therefore, the two algorithms have the same message complexity.

**E. Proof of Proposition 5**

**Proposition 5:** In DFAS, a packet is guaranteed to be forwarded \( 2h \) hops per \( (4h - 4) \) sub-epochs, irrespective of the flow direction and the origin sensor.

**Proof:** To prove Proposition 5, we apply DFAS on a path of length \( 4h - 3 \) nodes (Fig. 2). The path length is selected to accommodate four subpaths \( B_1, B_2, B_3 \) and \( B_4 \), so that the proposition can be proved, irrespective of the sensor that originates a packet transmission and the traffic direction. Without loss of generality, we select \( v_1 \) to be the pseudo-sink and label each node in the path according to Step 3 of Algorithm 8. We then allocate the transmission times for each node according to Step 4 of Algorithm 8. The sub-epoch assignments for each node are shown in Fig. 2. The arrows between sub-epochs show the possible flow of traffic in two consecutive sub-epochs, in either direction.

First, consider nodes with ids 1 or \( h \). From Fig. 2, it is straightforward to verify that a packet \( m \) originating from a node with id 1 or \( h \) can be forwarded in the upstream or the downstream direction over

\[
H_1 = 4h - 4 \text{ hops}
\]

in \( 4h - 4 \) sub-epochs. This is because the neighbors of nodes with id 1 or \( h \) are assigned to transmit in successive sub-epochs in either direction. Consider now a node from \( B_1 \) or \( B_3 \) with id other than 1 or \( h \), say \( j \), which is assigned to transmit during sub-epochs \( I_{j}, I_{2h-j}, I_{2h+j-2} \), and \( I_{4h-j-2} \). For the upstream direction, the worst-case delay for node with id \( j \) is incurred when it initiates the packet forwarding in sub-epochs \( I_{2h-j} \) or \( I_{4h-j-2} \). This is because the upstream neighboring node of \( j \) is scheduled to transmit \( 2j - 2 \) sub-epochs after \( j \) (as opposed to being scheduled in the successive sub-epoch when \( j \) initiates the upstream forwarding in \( I_{j} \) or \( I_{2h+j-2} \)). Once the packet reaches the upstream neighboring node of \( j \), it can be relayed \( 4h-4-(2j-2) = 4h-2j-2 \) hops in the remaining sub-epochs of the \( 4h-4 \) period. Thus, the total number of hops relays for node with id \( j \) is equal to

\[
H_2 = 4h - 2j - 2 \text{ hops},
\]

which obtains the minimum value of \( H_2 = 2h \) hops for \( id = h - 1 \). In a similar way, for the downstream direction, the worst-case delay for node with id \( j \) occurs when it starts forwarding \( m \) in sub-epochs \( I_{j} \) or \( I_{2h+j-2} \). The downstream node of \( j \) is now scheduled to transmit \( 2h - 2j \) sub-epochs after \( j \). After the packet reaches the downstream neighboring node of \( j \), it can be relayed \( 4h - 4 - (2h - 2j) = 2h + 2j - 4 \) hops in the rest of the \( 4h - 4 \) period. The total number of hops for node with id \( j \) is

\[
H_3 = 2h + 2j - 4 \text{ hops},
\]

which obtains its minimum value of \( H_3 = 2h \) hops for \( id = 2 \). Note that, if the node was chosen from \( B_2 \) or \( B_4 \), the same analysis applies with the difference that \( H_2 \) would correspond to the downstream case, and \( H_3 \) to the upstream one.

Finally, the minimum number of hops irrespective of the flow direction, in a period of \( (4h - 4) \) sub-epochs is equal to

\[
H_{\text{min}} = \min\{H_1, H_2, H_3\} = 2h \text{ hops},
\]

for \( h \geq 2 \).

**F. Proof of Proposition 6**

**Proposition 6:** Let a sensor \( v \) belong to \( f(v) \geq 1 \) CDSs. Suppose that an event \( \Psi \) is detected by \( v \) at time \( t(\Psi) \), where \( t(\Psi) \) is uniformly distributed over \( z \) epochs. The delay until a CDS containing \( v \) becomes active is:

1) \( d_{\text{min}} = 0 \) epochs.
2) \( d_{\text{max}} = z - f(v) \) epochs.
3) \( d_{\text{ave}} = \sum_{k=1}^{z-f(v)} k \times \frac{C(z-k-1,f(v)-1)}{C(z,f(v))} \) epochs.

**Proof:** Without loss of generality, assume that event \( \Psi \) observed by \( v \) occurs during the first epoch. If \( v \in D_1 \), the event is reported during this epoch and the buffering delay equals zero. This defines the minimum possible rotation delay \( d_{\text{min}} \). If \( v \in D_2 \) and \( v \notin D_1 \), then \( v \) must wait for one epoch before it can transmit the event report to the sink. This occurs with probability,

\[
Pr[d = 1] = \frac{C(z-2, f(v)-1)}{C(z, f(v))},
\]

where \( C(n,k) \) denotes the binomial coefficient. The numerator in (1) denotes the number of ways that \( f(v) \) elements (appearance frequency of \( v \) in the \( z \) CDSs) can be chosen from \( z \) elements (number of CDSs), such that one specific element (in our case, \( D_2 \)) is always selected, while another (i.e., \( D_1 \)) is always not selected. Alternatively, this can be seen as eliminating \( D_1 \) from

1. For simplicity, we have ignored the case in which sensor \( v \) has already transmitted all its dummy packets during the first epoch, before \( \Psi \) has occurred.
f(v) \in D_1, \ldots, D_z$ fixing $D_2$ and selecting $f(v) - 1$ subsets from the remaining $z - 2$ subsets. The denominator denotes the number of ways that $f(v)$ elements can be chosen from $z$ elements. In the computation of the probability in (1), we have implicitly assumed that all arrangements of $f(v)$ elements out of $z$ are equiprobable.

Generalizing to the case in which $v$ has to wait for $k$ epochs,

$$\Pr[d = k] = \frac{C(z - k - 1, f(v) - 1)}{C(z, f(v))}. \quad (2)$$

In the worst-case scenario, sensor $v$ will wait for $z - f(v)$ epochs before one of the subsets that contain it becomes active. This scenario occurs if $v$ is part of the remaining subsets $D_{z - f(v) + 1}, \ldots, D_z$. The value of $z - f(v)$ yields the worst case rotation delay for $v$. To compute the average delay, we sum over all possible delay values, multiple by the respective probability for each delay.

$$d_{ave} = \sum_{k=1}^{z-f} k \times \frac{C(z - k - 1, f(v) - 1)}{C(z, f(v))}.$$  

G. Proof of Proposition 7

**Proposition 7:** The number of hops traversed by a real packet $m$ originating from $v$ until it reaches the sink $\mu$ is upper-bounded by $|p(v, \mu)| + 2\text{rot}$, where $|p(v, \mu)|$ is the shortest path length between $v$ and $\mu$ and $\text{rot}$ is the number of CDS rotations until $m$ reaches $\mu$.

**Proof:** Without loss of generality, let $D_1$ be active during epoch $W_k$. Node $v \in D_1$ initiates the relay of a real packet $m$ to the sink $\mu$. Assume that $w \in D_2$ is the last node to forward $m$, before rotating to a CDS $D_2$ in $W_{k+1}$. The transmission of $w$ is overheard by a node $u \in D_2 \cap N_w$. First, we show via contradiction that paths $p(v, w)$ and $p(w, \mu)$ are also shortest paths between the respective sources and destination. We break the shortest path $p(v, \mu)$ into subpaths $p(v, w)$ and $p(w, \mu)$ and assume that $p(v, w)$ is not a shortest path between $v$ and $w$. That is, there is a path $p^*(v, w)$ such that $|p^*(v, w)| < |p(v, w)|$. Hence, there exist a path $p^*(v, \mu) = p^*(v, w) \cup p(w, \mu)$ with $|p^*(v, \mu)| < |p(v, \mu)|$, which contradicts the assumption that $p(v, \mu)$ is the shortest path.

With the termination of the epoch $W_k$, packet $m$ has traversed $|p(v, w)| + 1$ hops to reach $u \in D_2$. ($|p(v, w)|$ to reach $w$ and one hop to reach $u$). The one-hop neighborhood of $w$ can be divided to three possible subsets with respect to their hop distance to the sink. One subset at a distance of $|p(w, \mu)|$, one subset at a distance $|p(w, \mu)| + 1$, and one subset at a distance $|p(w, \mu)| - 1$, where $|p(w, \mu)|$ is the length of the shortest path between $w$ and $\mu$. Again, this claim can be shown via contradiction. If there was a neighbor $k \in N_w$ that could reach $\mu$ at fewer hops than $|p(w, \mu)| - 1$, then $w$ has a shorter path to $\mu$ via $k$, which contradicts the assumption that $p(w, \mu)$ is the shortest path. If there was a neighbor $k \in N_w$ with a shortest path to $\mu$ longer than $|p(w, \mu)| + 1$ hops, then $k$ could reach $\mu$ in fewer hops via $w$, which contradicts the fact that every CDS contains shortest paths to $\mu$.

We therefore conclude that in Step 2 of Algorithm 9, when node $u \in D_2 \cap N_w$ continues to forward $m$ to $\mu$ via the shortest path, $p(u, \mu)$ has one of the following sizes:

- $|p(u, \mu)| = |p(w, \mu)| - 1$
- $|p(u, \mu)| = |p(w, \mu)|$
- $|p(u, \mu)| = |p(w, \mu)| + 1$

The worst case occurs when $|p(u, \mu)| = |p(w, \mu)| + 1$. If $m$ is delivered during $D_2$, packet $m$ traverses in the worst case

$$H = |p(v, w)| + 1 + |p(u, \mu)| + 1 = |p(v, \mu)| + 2$$

Therefore, the number of hops traversed by $m$ due to the rotation of CDSs, increases up to 2 hops. The same process is repeated each time that a CDS rotation occurs. So in the worst case, the total number of hops traversed by $m$ is upper-bounded by $|p(v, \mu)| + 2\text{rot}$, where $\text{rot}$ is the total number of rotations required to delivered the packet.

**REFERENCES**
