Ensuring Cybersecurity via. Distributed On-line Secret Sharing

Secure State Recovery of Unmanned Aerial Vehicles from Adversarial attacks

Bharath Satheesh
Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, California, USA
bharath.satheesh@berkeley.edu

***Abstract*—The next decade is going to be witnesses to a large influx of unmanned aerial vehicle (UAVs). Many companies and institutions have already predicted the use of tens of thousands of UAVs for different purposes from grocery delivery to surveillance, Internet connectivity to recreational flying. So then an important question becomes, how does one go about ensuring safety and robustness for these cyber physical systems (CPS) against adversarial cyber attacks. The fewer the assumptions made about the attacker, the larger the set of attacks that a method can help the UAV system recover. This paper effectively demonstrates how an extension of concept of secret sharing from coding theory can be ported here to create efficient and effective recovery of state messages between any two on-line, dynamic systems. Next, the paper proposes how this method of secret sharing can naturally be extended and scaled with the help of an extension to Dijkstra's algorithm to accommodate a larger UAV traffic while effectively lowering bandwidth costs and maintaining an optimal level of autonomy. Finally, the paper provides reasonable robustness guarantees on secure state recovery under an adversarial cyber attack and provides an approximate, optimal time for recovery.**

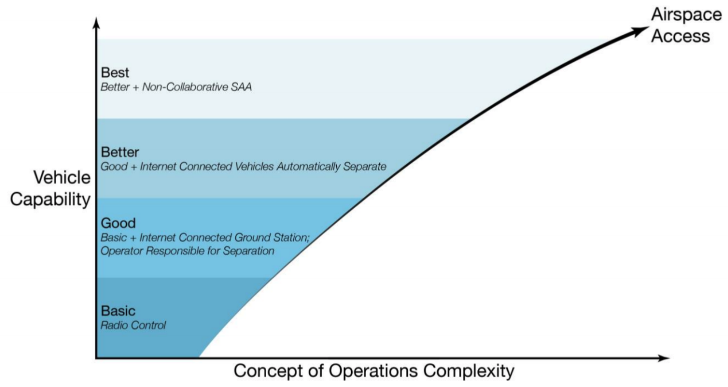

Airspace Access
Best
Better + Non-Collaborative SAA
Better
Good + Internet Connected Vehicles Automatically Separate
Vehicle Capability
Good
Basic + Internet Connected Ground Station; Operator Responsible for Separation
Basic
Radio Control
Concept of Operations Complexity

Fig. 1. *The following model put forward by Amazon[1] at the UTM NASA conference (UTM NASA 2015) shows the amount of autonomy that UAVs are required to maintain in the airspace at all times.*

I. Introduction

As the popularity of Unmanned Aerial Vehicles (UAVs) grow at a tremendous rate over the next decade, many large companies like Amazon, Facebook, Google etc. want to use these UAVs to perform various tasks including package delivery [1], Internet connectivity [2], surveillance, agriculture etc. leading to widespread civilian applications that have the potential to provide great benefits to society. Given this popularity, an important question that needs to be answered at this stage is one of ensuring security against cyber attacks. According to the specifications laid out by the FAA, NASA and other players [3], maximum autonomy in communication is of utmost importance. This means that quadrotors(UAVs) would in general communicate state estimates routinely to:

- To communicate with a central base station to provide GPS estimates/ change in route to accommodate increase in UAV traffic during peak hours
- To communicate with another UAV to provide state estimates in order to perform collision avoidance manoeuvres when the UAVs get too close (as detected by sensors onboard the UAV)

These communication links could be subject to cyber attacks of various forms. Various researchers have studied these attacks and in general classify them them into 3 main categories. In the Crdenas et al. paper [4] the authors classify traditional security goals for cyber physical systems into *integrity, confidentiality and availability*. Of the main forms, I picked an attack called the Man-in-the-Middle attack (MITM) because it encompasses parts of each of the four main classes of attacks since the attacker in this case has the ability to spoof messages, listen in/tap on incoming and outgoing messages and jam signals and communications that lead to denial of service (DDoS) errors.

Researchers have studied various approaches to securing general cyber-physical systems, each based on a different set of assumptions about the attacker. For example, the authors in [6], [7] assume that the attack signal would follow certain probabilistic distributions and then design filters for detection of such attacks. In [8], [9], [10], [11], [12], the authors use the game theory framework, where the controller and attacker are players with competing goals in a game. Attackers are assumed to adopt specific strategies that maximize a certain cost and the controller or estimator is designed to minimize such a cost. More recently, Fawzi et al. proposed in [13] a secure estimation method for arbitrary attacks, with a limiting assumption that the set of attacked sensors do not change with time. In this paper, we focus on sensor attacks on UAVs and attempt to design a secure estimator for linear time-invariant (LTI) systems based on as few assumptions about the attackers as possible. More recently, Qie et al. [5] does not assume that the attack signals follow any stochastic

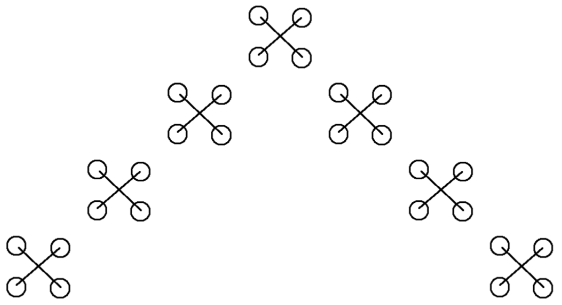


Fig. 2. *The platoon configuration of the UAV system with one leader and six followers in the platoon on an air highway.* ***Note:*** *This implementation is different from the traditional implementation in the paper by Qie et al.[5] where the platoon configuration is a straight line*

distributions, and thus their proposed secure estimator + Kalman filter model works for arbitrary and unbounded attacks and allows the set of attacked sensors to change over time. The only assumption made is on the sparsity of the attacked sensors at any given time step.

In this paper, there are no assumptions about the sparsity of the attacked signals or of the attacker, thus making the implementation more robust and practical for a large scale, real-world system. Further, the only assumption that this paper makes is that the number of attackers are sparse with respect to the number of actual UAVs in the air at any point. This assumption is a fair one since we can guarantee this scenario with extremely high probability given the on-line nature of the problem and the assumptions that we would have thousands of UAVs in the air. Once we can provide a satisfactory solution that guarantees secure state recovery for a small number of UAVs, then we can easily convert the secret sharing problem into a shortest path graphing problem in order to provide bounds for cyber security in large UAV networks.

II. A Preliminary Overview

A. Secret Sharing Schemes

The traditional Shamir's secret sharing scheme [14] divides data D into n pieces in such a way that D is easily reconstructable from any k pieces, but even complete knowledge of $k-1$ pieces reveals absolutely no information about D. This technique enables the construction of robust key management schemes for cryptographic systems that can function securely and reliably even when misfortunes destroy half the pieces and security breaches expose all but one of the remaining pieces. The traditional scheme is based on polynomial interpolation: given k points in the 2-dimensional plane $(x_1, y_1,).....(x_k, y_k)$. with distinct x_i's, there is one and only one polynomial $q(x)$ of degree $k-1$ such that $q(x) = y_i$ for all i. Without loss of generality, we can assume that the data D is (or can be made) a number. To divide it into pieces D , we pick a random $k-1$ degree polynomial $q(x) = a_0 + a_1 x + ... + a_{k-1} x^{k-1}$ in which $a_0 = D$, and evaluate:

$$D_1 = q(1).....D_i = q(i).....D_n = q(n). \tag{1}$$

Given any subset of k of these D_i values (together with their identifying indices), we can find the coefficients of $q(x)$ by interpolation, and then evaluate $D = q(O)$. Knowledge of just $k-1$ of these values, on the other hand, does not suffice in order to calculate D.
In further extensions of Shamir's Secret-Sharing scheme such as SNEAK[15], the dealer only distributes the secrets once to a certain fraction of the total population of the system. The Algorithm manages to redistribute new combinations of the secret to new members of the group such that there is lower bandwidth cost and information is reused.We can see that SNEAK is completely distributed requiring knowledge of only one-hop neighbours as opposed to the SMT-based solution which requires the knowledge of the global topology in order to set-up communication over node-disjoint paths. Also, SNEAK requires a communication of only 12 values, as opposed to 24 under the SMT-based solution. The number of random values generated under SNEAK is only 2, whereas the SMT-based solution requires generation of 5 random values

B. Asymmetric Encryption: Introduction to Security

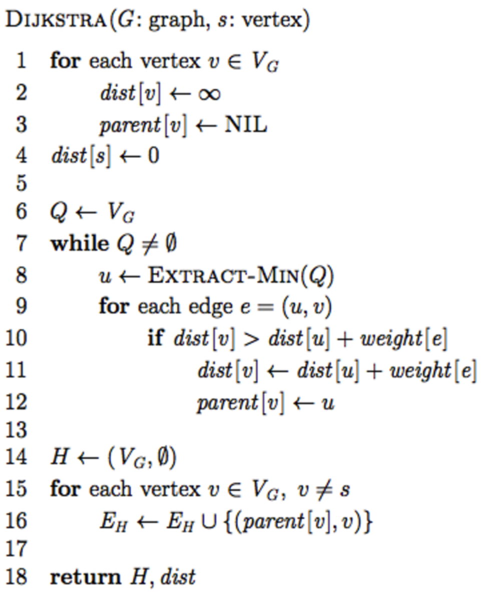
Public-key, or asymmetric, cryptography also emerged in the mid-1970s[16]. Digital signatures are based on asymmetric cryptography and can provide assurances of evidence to origin, identity and status of an electronic document, transaction or message, as well as acknowledging informed consent by the signer. This method can be utilized to protect messages from different ***integrity*** attacks. Asymmetric Encryption is a form of encryption where the different players in the system have 2 keys:

- A **public key** that is distributed publicly to anybody in the system that wants to send a message.
- A **private key** that is used to decrypt messages encrypted with the public key; nobody else has access to the private key.

To create a digital signature, signing software (such as an email program) creates a one-way hash of the electronic data to be signed. The user's private key is then used to encrypt the hash, returning a value that is unique to the hashed data. The encrypted hash, along with other information such as the hashing algorithm, forms the digital signature. Any change in the data, even to a single bit, results in a different hash value. This attribute enables others to validate the integrity of the data by using the signers public key to decrypt the hash. If the decrypted hash matches a second computed hash of the same data, it proves that the data hasn't be tampered with, after it was signed. If the two hashes (signatures) don't match, the data has either been tampered with in some way (indicating a failure of integrity) or the signature was created with a private key that doesn't correspond to the public key presented by the signer (indicating a failure of authentication).

C. Graphing Algorithms: The Shortest Path Problem

In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such



```
DIJKSTRA($G$: graph, $s$: vertex)
for each vertex $v \in V_G$
    $dist[v] \leftarrow \infty$
    $parent[v] \leftarrow$ NIL
$dist[s] \leftarrow 0$

$Q \leftarrow V_G$
while $Q \neq \emptyset$
    $u \leftarrow$ EXTRACT-MIN($Q$)
    for each edge $e = (u, v)$
        if $dist[v] > dist[u] + weight[e]$
            $dist[v] \leftarrow dist[u] + weight[e]$
            $parent[v] \leftarrow u$

$H \leftarrow (V_G, \emptyset)$
for each vertex $v \in V_G,\ v \neq s$
    $E_H \leftarrow E_H \cup \{(parent[v], v)\}$

return $H, dist$
```

that the sum of the weights of its constituent edges is minimized. The problem of finding the shortest path between two intersections on a road map (the graph's vertices correspond to intersections and the edges correspond to road segments, each weighted by the length of its road segment) may be modelled by a special case of the shortest path problem in graphs. Some of the possible graphing solutions to this problem are as follows:

- Bellman-Ford Algorithm
- Dijkstra's Algorithm
- A* Search Algorithm

In this paper, I consider the Dijkstras algorithm since it enables us to solve the single-source shortest path problem, in which we have to find shortest paths from a source vertex v to all other vertices in the graph. Further, Dijkstra's algorithm has a better run-time than the Bellman-Ford Algorithm and is more generalizable than the A* algorithm that is based more on graph heuristics.
For sparse graphs, that is, graphs with far fewer than $|V|^2$ edges, Dijkstra's algorithm can be implemented more efficiently by storing the graph in the form of adjacency lists and using a self-balancing binary search tree, binary heap, pairing heap, or Fibonacci heap as a priority queue to implement extracting minimum efficiently. To perform decrease-key steps in a binary heap efficiently, it is necessary to use an auxiliary data structure that maps each vertex to its position in the heap,, and to keep this structure up to date as the priority queue Q changes. With a self-balancing binary search tree or binary heap, the algorithm requires $\Theta((|E| + |V|) log|V|)$ time in the worst case

III. Implementation

A. Base Case

Suppose we have 4 UAVs in the sky close to one-another such that they are moving along the same direction. Suppose the proximity sensor on one of the UAVs (say UAV-C) detects that another UAV (say UAV-A) is dangerously close to minimum distance to separation between the two UAVs. Now, C requests state information from A so that it can perform a collision avoidance maneuver[]. However, the problem arises when A broadcasts its state information to C. Any nearby *honest-but-curious* UAVs nearby can extract information thus leading to **confidentiality attacks**. However, the worst case scenario occurs when there exists a Man in the Middle M that spoofs signals sent from A to C and vice versa.
This situation is described in *Fig 2* where A and C as well as two other neighbours B and D are shown. Now, to prevent confidentiality attacks, this paper picks up standard textbook Asymmetric encryption with both A and C having their own set of public and private keys. This way neither the neighbours B and D nor the Man in the Middle (M) can understand state information.

However, to protect this information transfer in the 4 UAV system, we need to guard against spoofing by the MITM, M. We refer to classical methods of secret sharing for this message transfer. Once A sends C a message and signature via. asymmetric encryption, C will be able to detect spoofing with the help of asymmetric verification methods and hence detect the presence of M in the AC channel. **Given that there are no MITM nodes between edges** AC, AD, BC **and** CD**,** A uses Shamir's classical secret sharing scheme to split up its state information and sends the packets to B and D. C then picks up the information from B and D to recover $A's$ state information and hence perform a collision-avoidance maneuver. This is depicted in the diagram shown as Fig. 3.

Why does this method work? The secret sharing scheme is a $(2, 2)$ scheme where the entire information is broken down into two parts and exactly two parts are required to recover the original information. As a consequence, the scheme is robust to collusion by individual nodes such as B and D. Another advantage is that the secret sharing scheme is optimal for the on-line nature of the problem where the UAVs in a given platoon are constantly changing and shifting.

B. Scaling Secret Sharing: Graphs

When we try and scale the secret sharing solution to a much larger platoon (i.e. 200 UAVs), we find that traditional methods will not suffice. As a result, we look forward to solving this state retrieval problem using graphing algorithms: especially the Dijkstra's Algorithm to find the shortest path between any two points in a given positively weighted graph G.

1) Choosing Weights: Weights in this graph signify the effective communication cost to transmit messages across the graph network. We note here that the weights between edges are assigned dynamically in this problem, since the weight between two nodes i and j, $w_{i,j}$ depends on the distance $d_{i,j}$

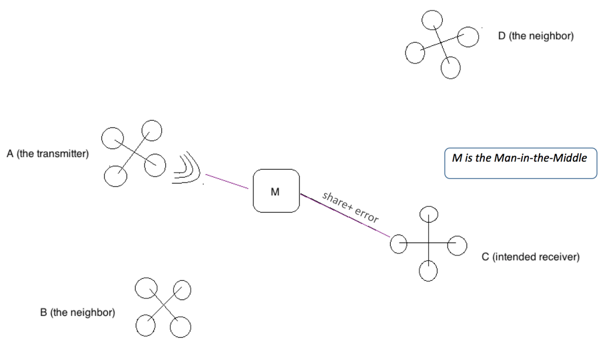


Fig. 3. *There are 4 UAVs on the air highway with 1 malicious Honest-but-curious node M between the UAVs. We see that two nodes can communicate with each other by simply encrypting messages with the help of Asymmetric Encryption.*

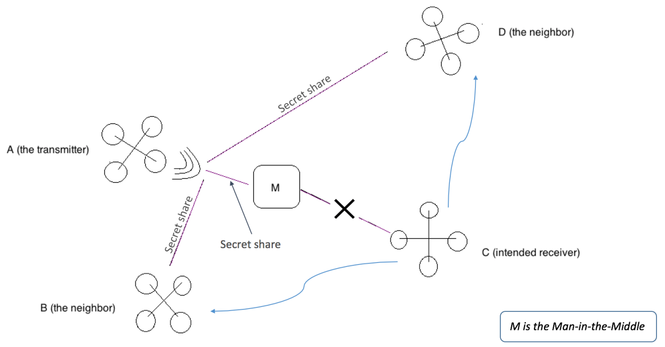


Fig. 4. *In this figure, M is a malicious MITM node that trys to spoof messages and communication between two UAVs. However, the UAV that transmits the node uses a $(2, 2)$ secret sharing scheme to split up its state information into 2 unique secrets such that any 2 UAVs (i.e. the receiver) can together recover the secret; This scheme is collusion resistant to any single node in the system*

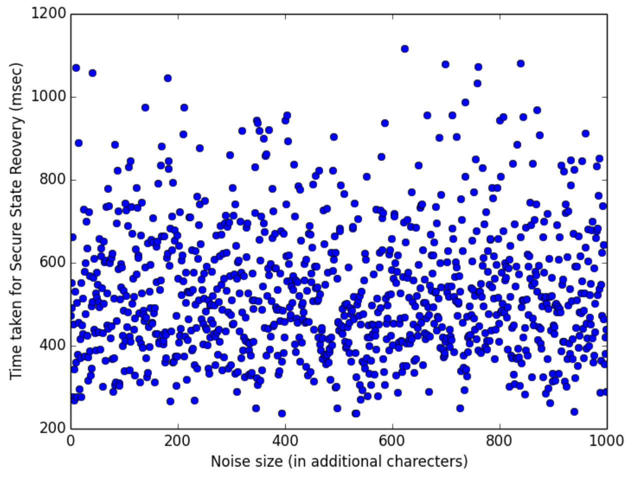


Fig. 5. *The plot shows us that as expected, regardless of the noise in the MITM channel, the secret sharing method recovers the state information in under 500ms. This time almost 100x faster than any current state recovery method via. Compressed sensing*

between the two nodes, the probability that the edge will be affected by an MITM attack $p_{i,j}$ and a constant $C_{i,j}$ that varies based on external factors such as weather condition, overall wind speed etc.

$$w_j \propto C_{i,j} d_{i,j} p_{i,j} \tag{2}$$

We will see that while performing Dijkstra's algorithm, we will update the weights of the graph edges at each step and hence vary our graph after every iteration. Further, we set the weights of an edge that received corrupted information to infinity so that that path is never traversed again. Dijkstra's Algorithm only requires that the weights of the edges be positive.

2) Dijkstra's Anti-Spoof Algorithm: This paper does not directly use the prepackaged Dijkstra's algorithm since for our $(2, 2)$ secret sharing scheme, the algorithm must work with twice starting along different edges so that the resulting end node (C) receives two shares of the secret from A, while all other neighbors only receive 1 share each. Dijkstra's algorithm initially marks the distance (from the starting point) to every other intersection on the map with $infinity$. This is done not to imply there is an infinite distance, but to note that those intersections have not yet been visited. Now, at each iteration, select the current intersection. For the first iteration, the current intersection will be the starting point, and the distance to it will be 0. For subsequent iterations (after the first), the current intersection will be the closest unvisited intersection to the starting point (this will be easy to find). Note here that if there is information corruption through this edge, then re-label that edge to have a new weight of $infinity$. Further, update the weights of the graph to show that every neighbor of the affected node has an edge distance of $infinity$ to avoid getting affected themselves

From the current intersection, update the distance to every unvisited intersection that is directly connected to it. This is done by determining the sum of the distance between an unvisited intersection and the value of the current intersection, and relabeling the unvisited intersection with this value (the sum), if it is less than its current value. In effect, the intersection is relabeled if the path to it through the current intersection is shorter than the previously known paths. Nodes marked as visited are labeled with the shortest path from the starting point to it and will not be revisited or returned to. This is especially useful since when we run the algorithm twice, simultaneously, we don't want to traverse the same node twice since then 1 node can put the two parts of the secret together to recover state information.

Continue this process of updating the neighboring intersections with the shortest distances, then marking the current intersection as visited and moving onto the closest unvisited intersection until you have marked the destination as visited. Once you have marked the destination as visited (as is the case with any visited intersection) you have determined the shortest path to it, from the starting point, and can trace your way back, following the arrows in reverse; in the algorithm's implementations, this is usually done (after the algorithm has reached the destination node) by following the nodes' parents from the destination node up to the starting node; that's why we keep also track of each node's parent.

Algorithm 1 Dijkstra's Anti-Spoof Algorithm

```
Input: $Graph(G), vertex_s(S), vertex_e(E)$
Output: $Distance - Dict(D), Predecessor - list(P)$
    Set $D, P, Q$(estimated Distances) to zero
    Set $Q$ of $vertex_s$ =zero
    Run 2 times
 1: for all vertices($v$) in $Q$ do
 2:     if v is equal to E then
 3:         break
 4:     end if
 5:     for all neighbors($n$) of a vertex do
 6:         if The edge between the $v$ and $n$ is infected then
 7:             $G[v][n], G[n][v] \leftarrow$ inf
 8:         else
 9:             Update step for regular Dijkstra
10:         end if
11:     end for
12: end for
13: return $D, P$
    =0
```

IV. Experimental Validation

A. Problem Setup

In order to fully verify and check that this method is in fact optimal, I compose a 23 edge weighted graph containing 11 UAVs with edge weights from the set $(2, 5, 4)$ such that a platoon like structure is formed by the graph. The hope is that even a small platoon can give us satisfactory results to then expand to a much larger platoon size. Further, I structure my graph such that all nodes have exactly 5 neighbors, except the platoon leader and the tail-UAVs which have 2 and 3 neighbors respectively. Further, I use the (2,2) secret sharing scheme to recover accurate state information from a corrupted network. On finally receiving secret shares, we apply a Kalman filter in order to de-noise the secret share of surrounding Gaussian noise from nature.

B. Algorithm Execution

1) Picking the Probability of Attack: In this example, we check the probability of successful recovery based on the number of UAV edges that are taken down at any given time step. We vary the probability of attack from 0 to 1 on the graph edges (i.e. like flipping a biased coin) and plot the probability of secure state recovery by the system.

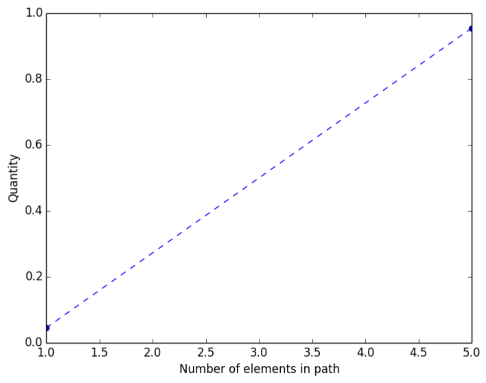


Fig. 6. *At most two out of 23 nodes are attacked;* 95% *recovery*

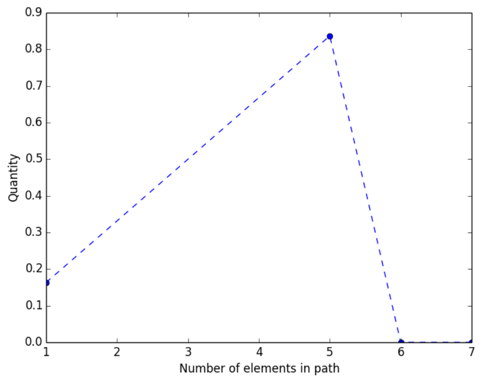


Fig. 7. *At most five out of 23 nodes are attacked;* 83% *recovery*

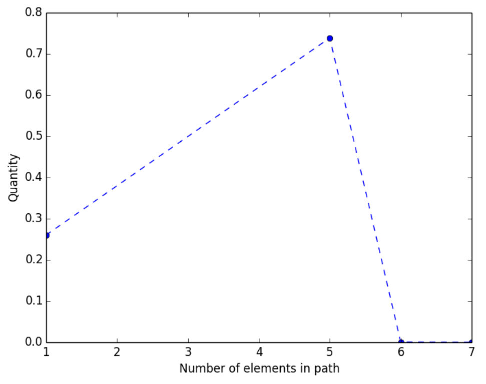


Fig. 8. *At most eight out of 23 nodes are attacked;* 75% *recovery*

2) Applying Asymmetric Encryption: UAVs A and C generate their own sets of public and private keys and A encrypts its state information with $C's$ public key and adds a signature with its own private key. As a result, all the UAVs will be able to verify that the message matches signature; however, only C will be able to recover the message with $C's$ private key. Further, if any other node, say M corrupts any information being sent, then C can identify quickly that the secret share has been corrupted. It then looks for another source to retrieve the secret shares.

3) Dijkstra's Algorithm: We run the modified Dijkstra's algorithm on our graph with edges being taken-down with a finite probability as described above. Then if a shortest path exists and was found, then we run the algorithm once again on the remaining set of vertices. If either fail to complete a run, then we return -1 and call our trial a failure.We try this process for 10000 iterations over each probability value to find the optimal probability of success under attack.

C. Results and Simulations

We see that when we increase the probability of attack uniformly[Fig 9], there is exponential decline in the number of successful recoveries. This makes sense because if more edges are down in the graph, there is a higher chance that the second Dijkstra will not reach the end-node C. Further, the Asymmetric Encryption takes a maximum of 650ms for state information encryption and transfer. As a result, we can say the following:

- As long as only a *reasonable* number of edges are attacked, there is an extremely high probability of recovery of state.
- There is an extremely high recovery rate given a certain proportion of edges are attacked in the graph G i.e. even a system where all the edges can be attacked at a time instances, this paper provides secure estimation 40% of the time.
- We see that the number of elements in the path are at most $5/6$ (i.e. each time the algorithm finds end vertex with at most 1 extra neighbor) which shows us that the graph design is near optimal
- The estimated time for state information retrieval is close to 620ms, which is almost 50x times faster than tradidional implementations/ state estimation methods
- By using a Kalman filter at the receiver, we robustify secret shares from corruption by Gaussian/ External (non-attack) noise.

V. Conclusion

In this project, I pose a secure state information retrieval problem for large UAV platoon-like systems and propose Secret Sharing and Asymmetric encryption as a method of protection against adversarial cyber attacks. Further, through simulation and experiments, this paper proves that this method guarantees state recovery with an extremely high probability based on the ratio of edges in a graph attacks w.r.t. the total number of nodes in graph showing that the platoon structure used is optimal. The exact design choice for the number of nodes in a platoon as well as the number of connections for each UAV in the platoon is left to the user's discretion. This is important especially since this system would work out cheaper, scale more efficiently and operate with near-complete autonomy in systems that we can expect in the coming decade.

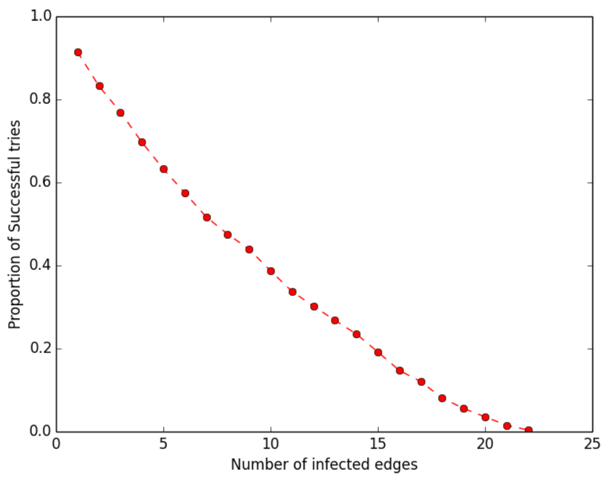


Fig. 9. *The number of successful trials decreases almost exponentially as the number of infected (attacked) nodes increases*

References

[1] Amazon Prime Air, http://www.amazon.com/b?node=8037720011.
[2] "Facebook Aquila Project" https://www.facebook.com/notes/mark-zuckerberg/the-technology-behind-aquila/10153916136506634/
[3] "UTM Nasa Conference,"https://www.nasa.gov/sites/default/files/atoms/files/utm-factsheet-11-05-15.pdf
[4] Alvaro Cardenas, Saurabh Amin, Shankar Sastry. "Secure Control: Towards Survivable Cyber-Physical Systems.". First International Workshop on Cyber-Physical Systems (WCPS2008)., June, 2008.
[5] Qie Hu, Young Hwan Chang, and Claire J. Tomlin, "Secure Estimation for Unmanned Aerial Vehicles against Adversarial Cyber Attacks," ICAS 2016 conference
[6] F. Pasqualetti, F. Dorfler, and F. Bullo, Cyber-physical attacks in power networks: models, fundamental limitations and monitor design, 50th IEEE conference on decision and control and european control conference, pp. 2195 2201, December 2011.
[7] K. Manandhar, X. Cao, F. Hu, and Y. Liu, Combating false data injection attacks in smart grid using kalman filter, International Conference on Computing, Networking and Communications, pp. 16 20, February 2014.
[8] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu, A survey of game theory as applied to network security, 43rd Hawaii International Conference on System Sciences, 2010.
[9] A. Gupta, C. Langbort, and T. Basar, Optimal control in the presence of an intelligent jammer with limited actions, 49th IEEE Conference on Decision and Control, pp. 1096 1101, December 2010.
[10] M. H. Manshaei, Q. Zhu, T. Alpcan, T. Basar, and J.-P. Hubaux, Game theory meets network security and privacy, ACM Computing Surveys, vol. 45, no. 3, June 2013.
[11] A. Gueye, V. Marbukh, and J. C. Walrand, Towards a metric for communication network vulnerability to attacks: A game theoretic approach, 3rd International ICST Conference on Game Theory for Networks, May 2012.
[12] M. Fei, M. Pajic, and G. J. Pappas, Stochastic game approach for replay attack detection, 52nd IEEE Conference on Decision and Control, pp. 1854 1859, December 2013.
[13] H. Fawzi, P. Tabuada, and S. Diggavi, Secure estimation and control for cyber-physical systems under adversarial attacks, Automatic Control, IEEE Transactions on, vol. 59, no. 6, pp. 14541467, June 2014.
[14] Adi Shamir, "How to Share a Secret," Communications of the ACM, 1979, Volume 22, Number 11
[15] Nihar B. Shah, K. V. Rashmi, Kannan Ramchandran, "Distributed Secret Dissemination Across a Network," IEEE Journal of Selected Topics in Signal Processing 9(7):1-1 October 2015
[16] "Asymmetric Encryption," http://searchsecurity.techtarget.com/definition/asymmetric-cryptography