AFRL-RI-RS-TR-2021-198

PREDICTIVE AND ADAPTIVE COMMUNICATIONS FOR REAL-TIME DYNAMIC NETWORK MANAGEMENT IN AUTONOMOUS UNMANNED AERIAL VEHICLE NETWORKS

NORTHERN ARIZONA UNIVERISTY

NOVEMBER 2021

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2021-198 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
ELIZABETH SERENA BENTLEY
Work Unit Manager

/ S /
GREGORY J. HADYNSKI
Assistant Technical Advisor
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
NOVEMBER 2021	FINAL TECHNICAL REPORT	FEBRUARY 2020	MAY 2021

4. TITLE AND SUBTITLE

PREDICTIVE AND ADAPTIVE COMMUNICATIONS FOR REAL-TIME DYNAMIC NETWORK MANAGEMENT IN AUTONOMOUS UNMANNED AERIAL VEHICLE NETWORKS

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8750-20-1-1000	N/A	62788F

5d. PROJECT NUMBER	5e. TASK NUMBER	5f. WORK UNIT NUMBER
T2TB	NO	AR

6. AUTHOR(S)

Dr. Fatemeh Afghah and Dr. Mohammed Gharib

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Northern Arizona University
School of Informatics Computing and Cyber Systems
1295 S. Knoles Dr
Flagstaff AZ 86011

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITGB
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

RI

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

AFRL-RI-RS-TR-2021-198

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this project, studied the communication and security challenges of UAV ad hoc networks from various aspects: i) We developed a new routing protocol for ad hoc UAV networks to handle the highly dynamic nature of such networks. We showed that the proposed routing algorithm outperforms all the well-known benchmarks in the flow success rate, throughput, and flow completion time; ii) we studied the security challenges of ad hoc UAV network and showed that existing pre-distribution-based key management protocols are vulnerable to cooperative attacks. We designed a Blockchain-based key exchange algorithm to increase the resiliency of the network against such attacks.

15. SUBJECT TERMS

Swarm networking, autonomous swarms, blockchain based key management

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	32

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
ELIZABETH SERENA BENTLEY	N/A

Contents

List of Figures

List of Tables

1 Summary

During this one year project, the team worked on two projects related to communication and security challenges of UAV ad hoc networks with the goal of handling the highly dynamic natures of these networks. First, we proposed a routing mechanism in ad hoc UAV network which considers the link lifetime and load balancing in addition to the traditional distance metric to develop a routing framework which finds the optimal path taking into the account future positions of the agent. Then, we studied a critical challenge in large-scale ad hoc UAV networks which is developing distributed security solutions and developed a key pre-distribution mechanism. Accordingly we describe each of the project in a separate section.

2 Routing in Ad Hoc UAV Network

2.1 Introduction

The cooperative ad hoc communication in UAV networks is of paramount importance, especially in the absence of sny infrastructure. Applications such as after disaster search and rescue and wildfire monitoring are some instances to name. To find the optimized route from the source UAV to the destination in such networks, the conventional routing algorithms such as Ad hoc On-Demand Distance Vector (AODV) and Dynamic Source Routing (DSR) are proposed to be used, which aim at finding the shortest path. Due to the highly dynamic nature of UAV movement in a three-dimensional sphere, we believe that the shortest path does not lead to the highest performance in terms of throughput, flow success rate, and flow completion time.

Accordingly, we defined the path lifetime metric as the shortest lifetime of the links forming the path. We proposed a link lifetime prediction algorithm to predict the link lifetime and consequently find the path lifetime for any possible path. We then proposed an optimization problem to find the optimized path considering both the path length and path lifetime at the same time. Obviously, the optimized path has to find the optimized trade-off between the path lifetime and path length to maximize the network performance in terms of throughput, flow success rate, and flow completion time. We managed the mentioned trade-off by weighting the path lifetime and path length more in the optimization problem. We further proposed a lightweight algorithmic method to find the solution of the optimization problem with the worst-case time complexity $O(|E|^2)$, where $|E|$ is the number of network links.

We performed a comprehensive evaluation for the proposed algorithms using the network simulator ns-3. We first designed and performed exhaustive simulations for different scenarios to find the optimized value for the weights of the optimization problem, for different network densities and different network loads. We then designed comparison scenarios to compare the proposed routing algorithm with a wide-range of well-known routing algorithms covering both distance vector and link state protocols, as well as proactive and reactive ones. In our performance evaluation process, we find that the proposed algorithm, referred to as optimized predictive adaptive routing algorithm (OPAR), outperforms all the compared routing algorithms in the throughput, flow completion time, flow success rate, and the generated routing traffic. It is worth mentioning that we developed an ns-3 code package

for our proposed routing algorithm which will be released soon. This package utilizes ns-3's different classes to initiate TCP connections between any pairs of source and destination nodes. It simulates different mobility models in three-dimensional space and lets the nodes move according to their chosen pattern. It finds the optimal path from the source to the destination according to the proposed OPAR routing algorithm. Hence, it first predicts the link lifetime for all network links. Then it solves the optimization problem with the proposed lightweight algorithm to find the optimal path. The developed package is completely compatible with any other ns-3 codes. Hence, any other ns-3 routing algorithms can easily take the OPAR's place for comparison reasons.

In our performance evaluation process, we found that the proposed OPAR, just like the other known routing algorithms, does not balance the network load. We further find that the load balancing could have a very high impact on the performance of the network, especially when the network's load is high, i.e. there are many concurrent communicating pairs. Hence, we proposed another optimization problem that considers the load of the paths as well as the path length and path lifetime in its route selection process. Obviously, the proposed algorithm has to work just like OPAR when we have just one communicating pair at a time. We further proposed a lightweight algorithmic solution to the new optimization problem and evaluated it through an exhaustive simulation using ns-3. Results show an obvious improvement in the investigated performance parameters.

The main contribution of this work is to define the path lifetime metric and to consider it besides the path length in finding the optimal path in highly dynamic ad hoc UAV networks. The link prediction in a 3D space, proposing a linear optimization problem to find the optimal path, developing a lightweight algorithmic solution for the optimization problem, and the exhaustive performance evaluations are the major contributions of this project. The proposed routing algorithm of this project is a potential candidate to be used in military applications where it is not feasible to use any communication infrastructure, or the infrastructure is damaged. We showed via OPAR how such optimization studies can improve the network performance.

2.2 Methods, Assumptions and Procedures

Assuming that we have a precise link lifetime prediction matrix $\mathcal{T}$, where $\tau_{(i,j)} \in \mathcal{T}$ is the lifetime of the edge $e_{(i,j)}$. Let us consider the variable $x_{(i,j)}$ as a Boolean variable specifying whether the link $e_{(i,j)}$ participates in the optimized path or not. On the one hand, the objective function of minimizing $\sum x_{(i,j)}$, such that the chosen edges form a path from the source to the destination, guarantees the shortest path. On the other hand, maximizing τ where τ is the path lifetime guarantees the path with the longest lifetime. If we design a multi-objective optimization problem that considers both of the mentioned metrics together, the solution will be an area of feasible solutions. Based on the priority of the objectives, the solution could be any points in this area. However, we designed the following Boolean linear programming (LP) that considers both metrics of interest, simultaneously. In the proposed optimization problem, parameters w_1 and w_2 are the weights defining the trade-off between the longer lifetime and shorter path where $w_1 + w_2 = 1$, $0 < w_1 \leq 1$, and $0 \leq w_2 \leq 1$. The higher value of these parameters leads to their higher impact on the optimization. The weights w_1 and w_2 are network setting parameters affected by many parameters, including

but not limited to, the network area, network density, transmission range, and network load. In this optimization problem, variable T represents the inverse of path lifetime, Constraints (9) and (10) guarantee that the source node has one outgoing edge and no ingoing edge, Similarly, Constraints (11) and (12) guarantee that the destination node has one ingoing edge and no outgoing. Constraint (13) guarantees that every vertex that has an ingoing edge has to have an outgoing edge, except for the source and the destination nodes. Constraint (14) is the main constraint that maximizes path lifetime. In the rest of this section, we describe how to find the solution to the proposed Boolean LP problem and how to form the matrix $\mathcal{T}$.

$$\min \quad \sum_{\substack{(i,j)\in E \\ i\in\{1,\cdots,n\} \\ j\in\{1,\cdots,n\}}} w_1 x_{(i,j)} + w_2 T$$

$$\text{Subject To:} \quad \sum_{(s,i)\in E} x_{(s,i)} = 1 \tag{1}$$

$$\sum_{(i,s)\in E} x_{(i,s)} = 0 \tag{2}$$

$$\sum_{(i,d)\in E} x_{(i,d)} = 1 \tag{3}$$

$$\sum_{(d,i)\in E} x_{(d,i)} = 0 \tag{4}$$

$$\sum_{\substack{(i,j)\in E \\ i\neq s}} x_{(i,j)} = \sum_{\substack{(j,k)\in E \\ j\neq d}} x_{(j,k)} \tag{5}$$

$$T \geq \frac{x_{(i,j)}}{\tau_{(i,j)}} \tag{6}$$

$$x_{(i,j)} \in \{0,1\} \tag{7}$$

$$0 \leq T \leq 1 \tag{8}$$

Optimization problem solution: The defined optimization problem is a Boolean LP (BLP), which is well-known to be an NP-complete problem. However, it is a special BLP case, where the following proposed algorithmic method can find its solution, i.e. the optimized path.

Assuming that we have a precise link lifetime prediction, we first sort the matrix of edge lifetimes in descending order. We then perform a breadth-first search (BFS) in the graph $G(V, E)$ to find the shortest path, without considering the link lifetimes. Let us refer to the shortest path as p_0. We then calculate the value of the objective function for the shortest path. Considering the link with the shortest link lifetime in path p_0 as $e_{(i',j')}$, we remove all the links with the lifetime lower or equal to the $\tau_{(i',j')}$. We then perform a BFS again, on the new graph and calculate the objective value for the new path. If the new objective value is less than the previous one, we replace the previous path with the new one and remove the edges with the lowest lifetime. We repeat this procedure until the BFS algorithm fails in

finding a new path. We show in Lemma (2.1) that the output of this algorithm is the path with the optimized objective function. We further show that the computational complexity of this algorithm is on the order $O(|E||V| + |E|^2)$, in Lemma (2.2).

Lemma 2.1. *Assume that there is at least one path from the source node to the destination. The output of the proposed Algorithm is the optimal path, i.e. the solution of the proposed Boolean LP problem.*

Lemma 2.2. *The computational complexity of the proposed algorithm is on the order* $O(|E||V|+|E|^2)$.

Now, we would like to add the load balancing information into the consideration. Hence, we assume that we have a matrix of all links' lifetimes, i.e. $\mathcal{T}$, as well as the vector of all nodes' load, i.e. $\mathcal{L}$. The objective function of this problem is to minimize the path length as well as path load, and maximize the path lifetime. The significance of each of the path length, lifetime, and load is set by setting the weights w_1, w_2, and w_3, respectively, where $\sum_{i=1}^{3} w_i = 1$. The variables representing the participants of network links in the optimal path are $x_{(i,j)}$s. If the link $e_{(i,j)}$ is selected to be a part of an optimized path, $x_{(i,j)}$ is set to one, elsewhere it is set to zero. In this objective function, T and L represent the inverse path lifetime, and the path load, respectively. While T has to be minimized according to the objective function, it has to be greater or equal to $\frac{x_{(i,j)}}{\tau_{(i,j)}}$ for all links $x_{(i,j)}$s, according to Constraint (14). The new method is to find the path lifetime, i.e. the minimum link lifetime for the links participating in the path, and consider it in the objective function. Similarly, the path load L has to be minimized. To find the path load, we minimize L in the objective function where in Constraint (16) we force L to be greater than or equal to $x_i l_i$, where according to Constraint (15) $x_i = \sum_{j=0, j\neq i}^{n} x_{(i,j)}$, and l_i is the network load of node i.

$$\min \quad \sum_{\substack{(i,j)\in E \\ i\in\{1,\cdots,n\} \\ j\in\{1,\cdots,n\}}} w_1 x_{(i,j)} + w_2 T + w_3 L$$

Subject To:

$$\sum_{(s,i)\in E} x_{(s,i)} = 1 \tag{9}$$

$$\sum_{(i,s)\in E} x_{(i,s)} = 0 \tag{10}$$

$$\sum_{(i,d)\in E} x_{(i,d)} = 1 \tag{11}$$

$$\sum_{(d,i)\in E} x_{(d,i)} = 0 \tag{12}$$

$$\sum_{\substack{(i,j)\in E \\ i\neq s}} x_{(i,j)} = \sum_{\substack{(j,k)\in E \\ j\neq d}} x_{(j,k)} \tag{13}$$

$$T \geq \frac{x_{(i,j)}}{\tau_{(i,j)}} \tag{14}$$

$$x_i = \sum_{\substack{j=0 \\ j\neq i}}^{n} x_{(i,j)} \tag{15}$$

$$L \geq x_i l_i \tag{16}$$

$$x_{(i,j)} \in \{0,1\} \tag{17}$$

$$0 \leq T \leq 1 \tag{18}$$

We need Constraints (9) and (10) to let the source node start the path and prevent it from being in a loop. Similarly, we need Constraints (11) and (12) to let the destination node be chosen as the last node in the path and terminate the path selecting operation. Constraint (13) assures that the intermediate node, which receives an active link, has to have an outgoing active link. As we discussed earlier, the proposed problem variables are Boolean, which is shown in Constraint (17). Finally, we need Constraint (18) in practice to remove the links with a very short lifetime. How short the link can be to be discarded depends on the metric chosen for link lifetime to be second, millisecond, etc. Algorithm (1) represents the solution of the proposed optimization problem with load balancing consideration.

Link lifetime prediction: While the proposed algorithm can generally work with any link lifetime prediction algorithm, we utilize three-dimensional geometry to predict the link lifetime by knowing the UAV nodes' positions. The optimization problem considers a $n \times n$ matrix $\mathcal{T}$ which contains the links' lifetimes as well as vector $\mathcal{L}$ with n elements which contain the nodes' load. In this section, we describe how we predict the link's lifetime and calculate the node's load to form $\mathcal{T}$ and $\mathcal{L}$. We start with $\mathcal{T}$ where we need only three consecutive node locations to fill it. Each node regularly sends a packet to the controller, including its last three consecutive locations in 3D space. The controller, in its turn, calculates the azimuthal and polar directions of each node as well as their speed and acceleration. Using

Algorithm 1 Finding the solution of the optimization problem

$objValue = \infty$, {Set the initial objective value equal to ∞.}
$path = \emptyset$, {Set the initial path to empty path.}
$\varsigma \leftarrow w_2\frac{1}{\mathcal{T}} + w_3\mathcal{L}$, {Form ς, a $n \times n$ matrix containing the lifetime-load cost of all graph edges. The lifetime-load cost for each edge $e_{(i,j)}$ is calculated as $\varsigma_{(i,j)} = w_2\frac{1}{\tau_{(i,j)}} + w_3 l_j$. }
$\varsigma_{Sorted} = descentSort(\varsigma)$, {Sort the lifetime-load cost matrix in descending order.}
$G'(V,E) = G(V,E)$, {Let $G'(V,E)$ equal to the main graph $G(V,E)$.}
while $isPath(G'(V,E), src, dst)$ **do**
  $newPath = BFS(G'(V,E), src, dst)$, {To find the shortest path with the BFS algorithm.}
  if $newObjValue < objValue$ **then**
    $path = newPath$,{Set the path equal to the new path.}
    $objValue = newObjValue$,{Set the objective value equal to the objective value of the new path.}
  end if
  $\varsigma_{path} \leftarrow lifetimeLoadCost(newPath)$, {Calculate the maximum path lifetime-load cost for the edges forming the $newPath$.}
  $G'(V,E) \leftarrow remove(\varsigma_{path}, \varsigma_{Sorted}, G'(V,E))$, {Remove all links with the lifetime-load cost greater than or equal to the ς_{path}.}
end while

this information, the controller calculates the time for each pair of neighboring nodes that they go out of the communication range of each other. To use simpler notations and without the loss of generality, we consider the communication range of all nodes equal and represent it with $\Re$. It is worth mentioning that our predicted lifetimes stay precise if the corresponding nodes do not change their direction or acceleration.

We represent the location of node u_i at time t by the tuple $(x_i(t), y_i(t), z_i(t))$. We show the distance traversed by the node u_i in the time interval $[t_1 \quad t_2]$ by $d^{(i)}_{(t_1,t_2)}$ and it could be easily calculated by Equation (19).

$$d^{(i)}_{(t_1,t_2)} = \sqrt{[x_i(t_2)-x_i(t_1)]^2+[y_i(t_2)-y_i(t_1)]^2+[z_i(t_2)-z_i(t_1)]^2} \tag{19}$$

Considering the three consecutive node locations at times t_0, t_1, and t_2, we calculate the azimuthal angle α_i and polar angle θ_i of the node's movement direction using Equations (20) and (21), respectively.

$$\alpha_i = tan^{-1}\left(\frac{y_i(t_2) - y_i(t_1)}{x_i(t_2) - x_i(t_1)}\right) \tag{20}$$

$$\begin{aligned}\theta_i &= cos^{-1}\left(\frac{z_i(t_2) - z_i(t_1)}{d^{(i)}_{(t_1,t_2)}}\right) \\ &= tan^{-1}\left(\frac{\sqrt{[x_i(t_2) - x_i(t_1)]^2 + [y_i(t_2) - y_i(t_1)]^2}}{z_i(t_2) - z_i(t_1)}\right)\end{aligned} \tag{21}$$

We can calculate speed v_i and acceleration a_i of node u_i respectively using Equations (22) and (23).

$$v_i(t_1) = \frac{\sqrt{d^{(i)}_{(t_0,t_1)}}}{t_1 - t_0} \tag{22}$$

$$a_i(t_2) = \frac{v_i(t_2) - v_i(t_1)}{t_2 - t_0} \tag{23}$$

Next, we can calculate the predicted location of the node u_i at time $(t_2 + \Delta t)$ using Equation (24). Then, we can calculate the distance between nodes u_i and u_j at time $(t_2 + \Delta t)$ using Equation (25). We represent this distance by $d_{(i,j)}(t_2 + \Delta t)$ where Δt is unknown. Knowing the communication range of nodes, i.e. $\Re$, we can calculate the time Δt using Equation (26). This variable is the time required for the nodes to go out of the communication range of one another and represents the lifetime of the link $e_{(i,j)}$. Equation (26) is a simple nonlinear equation with only one unknown variable, i.e. Δt. This equation can be easily solved with any lightweight numerical method such as the Bisection method. It is worth mentioning that the LB-OPAR is a modular routing algorithm. Hence, the prediction algorithm proposed here could be replaced with any other prediction algorithm with no need for any modification in the overall routing solution.

$$\begin{cases} x_i(t_2 + \Delta t) = x_i(t_2) + (v_i(t_2)\Delta t + \frac{1}{2}a_i\Delta t^2)sin(\theta_i)cos(\alpha_i) \\ y_i(t_2 + \Delta t) = y_i(t_2) + (v_i(t_2)\Delta t + \frac{1}{2}a_i\Delta t^2)sin(\theta_i)sin(\alpha_i) \\ z_i(t_2 + \Delta t) = z_i(t_2) + (v_i(t_2)\Delta t + \frac{1}{2}a_i\Delta t^2)cos(\theta_i) \end{cases} \tag{24}$$

$$d^{(i,j)}_{(t_2+\Delta t)} = \sqrt{\begin{array}{l}(x_i(t_2 + \Delta t) - x_j(t_2 + \Delta t))^2 + \\ (y_i(t_2 + \Delta t) - y_j(t_2 + \Delta t))^2 + \\ (z_i(t_2 + \Delta t) - z_j(t_2 + \Delta t))^2\end{array}} \tag{25}$$

$$d^{(i,j)}_{(t_2+\Delta t)} - \Re = 0 \tag{26}$$

To calculate each node's load, the controller starts the network operation with all zeros for the node's load. Any time the controller receives a route request, it calculates the optimal path and determines the nodes affected by the new route. Any node included in the new route and all its one hop neighbors are considered as the nodes affected by the new route. Since the controller potentially knows all nodes' locations, it can also calculate the distance between each pair of nodes, and it further knows the communication range of each node. Hence the controller can determine the affected nodes by the new route. The controller then increases the load of the determined nodes by one. Every time that a route becomes unavailable, the controller decreases the affected nodes' load by one. In two cases, a route is considered unavailable. First, the source node does not receive the TCP acknowledgment packets, and consequently, the source node orders a reroute. Second, the source node receives the acknowledgment of the last packet of the file and sends a flow termination message to the controller.

Table 1: Simulation Setting

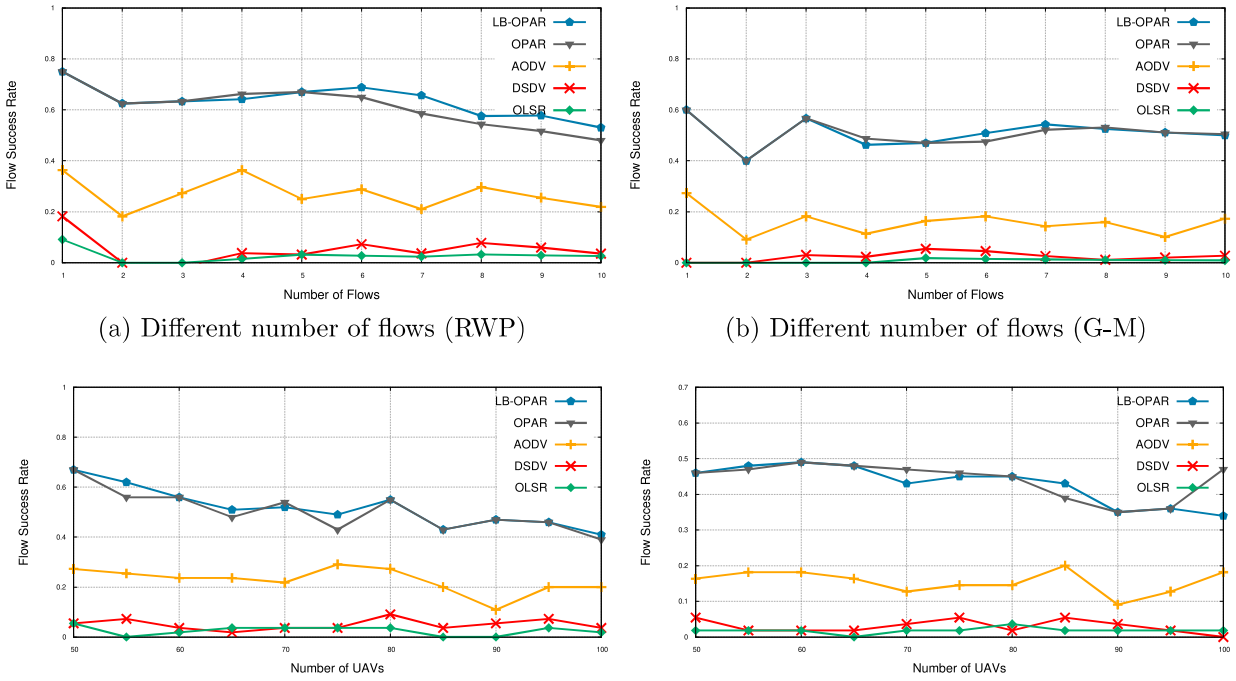
Simulator version	ns-3 3.30
Number of UAVs	[50 100]
Area size	$300 \times 2000 \times 50m$
Transmission power	7.5 dBm
Number of concurrent flows	[1 10]
File size	5 MB
Simulation time	500 sec
Speed range	$[0 \quad 50]m/s$
Mobility models	3D RWP
Traffic type	TCP NewReno
Wireless communication standard	IEEE 802.11b
Propagation loss model	Free-space propagation loss
Propagation delay model	Constant speed propagation delay

2.3 Results and Discussions

To evaluate the performance of the proposed algorithm, we use the ns-3 network simulator. Ns-3 is a well-known discrete-event simulator widely used for research and development. We simulate the network for different scenarios with a different number of nodes and a different number of concurrent file-transfer flows to exhaustively evaluate the network performance in different situations. Network throughput, flow completion time, and flow success rate are measured as the performance evaluation metrics. To show the significant improvement of LB-OPAR in the network performance, we compare its results with the baseline benchmarks AODV, OLSR, DSDV as well as the OPAR algorithm. We compare the results for flow success rate, network throughput, and flow completion time for different network densities and different network loads, i.e. concurrent flows. Table (1) represents the simulation setting.

Fig. (1) shows the results for the flow success rate. DSDV and OLSR routing protocols show less than 10% of flow success rate, in most cases, which is mainly because of their inability in catching up with the high dynamicity of the nodes' movement in 3D space. AODV, however, represents better performance in terms of flow success rate in comparison with DSDV and OLSR. It successfully delivers 30% of the flows, in average. Indeed, both OPAR and LB-OPAR significantly outperform the other algorithms by the average of 60% flow success rate. It is obvious that increasing the network load leads to more significant performance improvement for LB-OPAR compared to OPAR by an average of 5%. For less concurrent flows, there is no significant different between LB-OPAR and OPAR since the network load weight is zero or close to zero, as opposed to the case when the network has five concurrent flows and different densities, i.e Fig. (1c) and (1d). Something worth mentioning is that AODV, DSDV, and OLSR show a higher success rate under the random waypoint (RWP) mobility model than Gauss-Markov (G-M). It seems that the randomness of the RWP model helps them to increase their performance. A similar fact was previously discovered in different network studies [1]. OPAR and LB-OPAR show no significant difference under RWP and G-M mobility models, signifying their improved performance in different conditions.

Next, we investigate the network throughput for all the combinations of different network



(a) Different number of flows (RWP)

(b) Different number of flows (G-M)

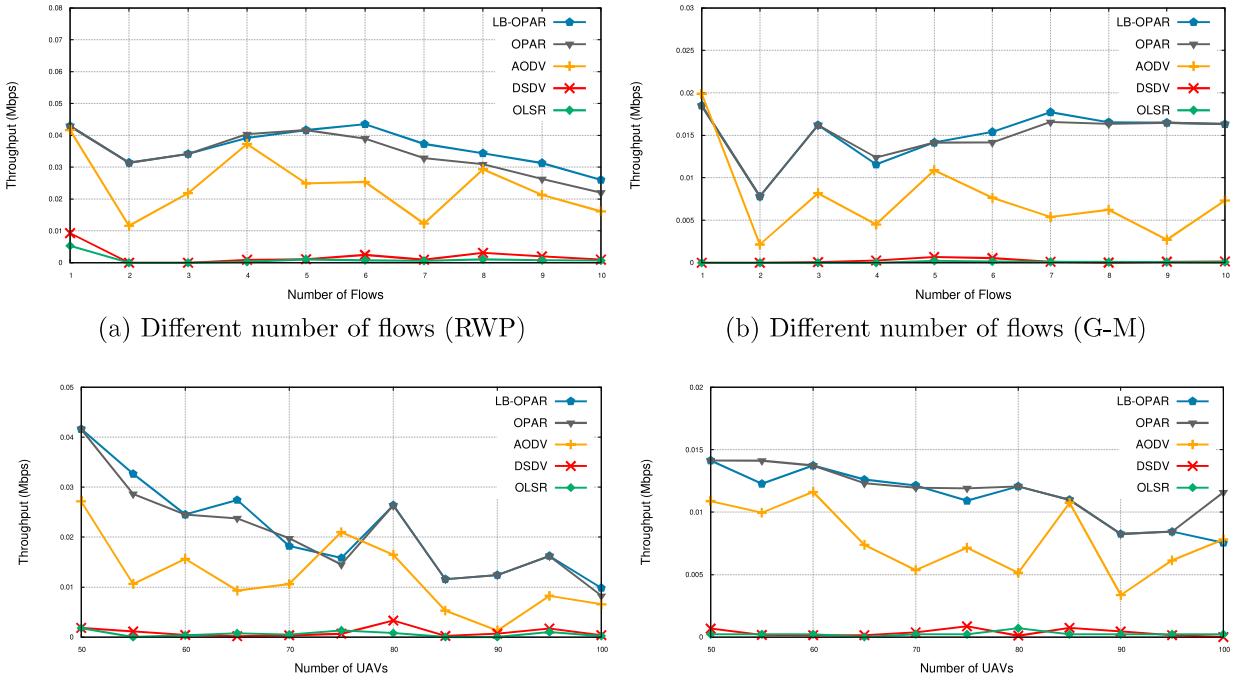
(c) Different number of UAVs (RWP)

(d) Different number of UAVs (G-M)

Figure 1: A comparison of the flow success rate for different loads and densities.

loads, network densities, and mobility models. Fig. (2) shows the results where the achieved throughput by DSDV and OLSR is close to zero due to their failure in handling the fast topology changes. Interestingly and in contrast to the conventional mobile networks, increasing the number of concurrent flows shows no significant effect on the throughput, which means in highly dynamic networks, the bottleneck of the throughput is the reroute process due to the topology change not the load of the network. For this parameter, AODV shows much better performance in comparison with DSDV and OLSR, while OPAR and LB-OPAR outperforms AODV by 100%. OPAR and LB-OPAR show competitive behavior. However, LB-OPAR outperforms OPAR in higher network loads and under the RWP mobility model by about 20%. For different network densities, since there are only five concurrent flows and the network load weight is zero or close to zero, we see no significant differences between LB-OPAR and OPAR. However, they outperform AODV by the average of 100%.

Fig. (3) shows the results for flow completion time. As we mentioned earlier, we consider the simulation time as the lower bound of the flow completion time (FCT) for the failed flows. Hence, the results of Fig. (3) show the lower bound for the different algorithms' FCT. This figure includes the combination of all network densities, loads, and the simulated mobility models. For all simulation settings, OPAR and LB-OPAR show the lowest FCT compared to the other routing algorithms. However, LB-OPAR for the network with higher loads shows around 15% less FCT in comparison with OPAR, under RWP mobility model. OPAR and LB-OPAR outperforms AODV by 10%, on average. AODV shows a significantly lower FCT than that of OLSR and DSDV where their average FCT is close to the simulation time, i.e. 500 s, due to theire high failure rate. In many cases, OPAR and LB-OPAR show



(a) Different number of flows (RWP)

(b) Different number of flows (G-M)

(c) Different number of UAVs (RWP)

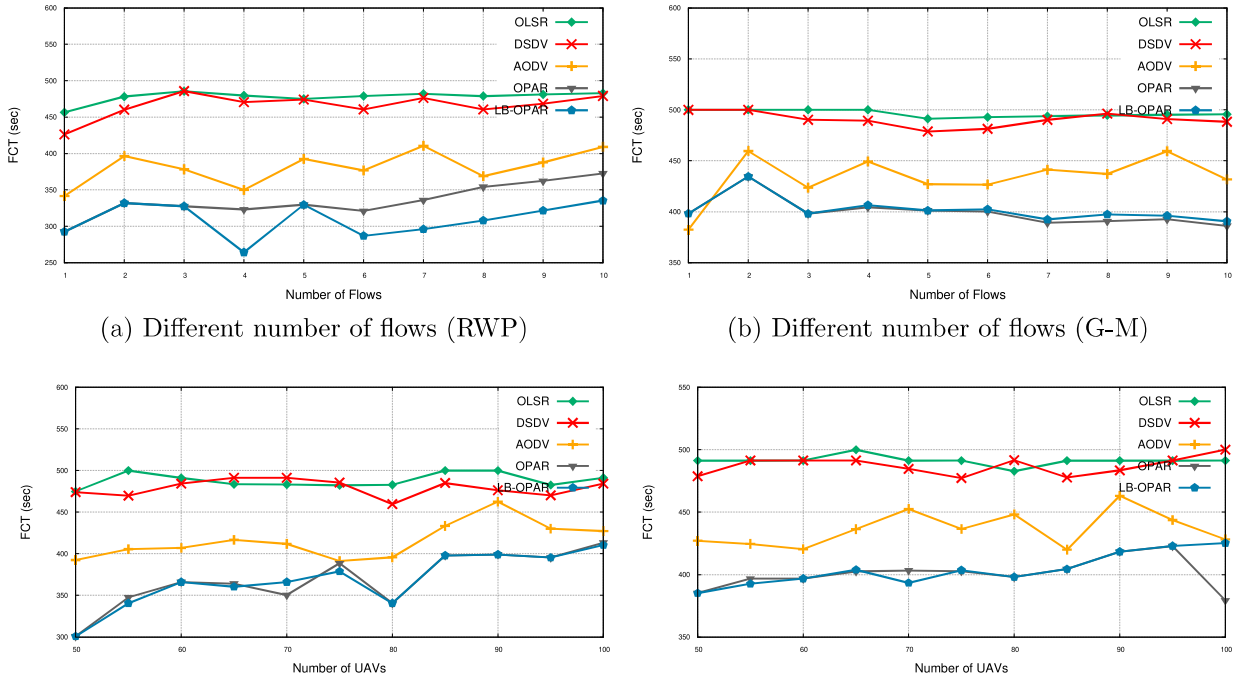
(d) Different number of UAVs (G-M)

Figure 2: A comparison of the throughput for different loads and densities.

close FCT results, which is one of two cases. The first case is when there a small number of concurrent flows which leads to a negligible effect on the network load in choosing the optimal path. The latter case owes itself to the fact that in some cases, there is only one path, or there are several paths, but just one of them has a reasonable objective value, where the others are very long or with a very short lifetime. Hence, both the OPAR and LB-OPAR select the same path, which leads to the close results.

2.4 Conclusions

The highly dynamic nature of cooperative UAV networks causes the conventional routing algorithms, which aim at finding the shortest path, to fail in reaching the optimal network performance. While we have shown, in our recently published work OPAR, that considering the path lifetime in selecting the routes has a significant impact on the network performance, we extend OPAR to consider the network load as well. Accordingly, we proposed load-balanced OPAR, an software defined networking (SDN)-based routing algorithm for cooperative UAV networks. We modeled the entire problem with an analytical model and proposed a lightweight solution for the proposed model. We implemented LB-OPAR in the ns-3 network simulator and exhaustively evaluated its performance. We show that LB-OPAR outperforms the benchmark routing algorithms AODV, DSDV, and OLSR. It also improves the performance of OPAR when the network works under higher loads. We further found that the performance bottleneck in highly dynamic UAV networks is basically the route breaks down, more than any other reason. Hence, proposing a routing algorithm



(a) Different number of flows (RWP)

(b) Different number of flows (G-M)

(c) Different number of UAVs (RWP)

(d) Different number of UAVs (G-M)

Figure 3: A comparison of FCT for different loads and densities.

to cover the multiple reroutes problem has significant importance in improving cooperative UAV networks' performance. Since LB-OPAR has a lightweight computational and space complexity, we aim at distributing the SDN controller tasks among all network nodes for fully distributed networks as future work.

3 Key Pre-Distribution in Ad Hoc UAV Networks

3.1 Introduction

Unmanned Terrestrial vehicles (UTVs) and unmanned aerial vehicles (UAVs) are increasingly deployed in a variety of applications such as disaster relief, military missions, terrain reconnaissance, traffic management, and fire detection. UTVs and UAVs, in infrastructure-free environments, use multi-hop cooperative networks as their underlay communication network. The security of the underlying wireless networks is critical in such highly sensitive operations. Alas, the lack of trusted infrastructures and limited node resources make securing communications in such networks challenging. While cryptography is a general and powerful approach to improve security, it is not well suited for such infrastructure-free networks. This is because cryptography techniques, such as public key infrastructure (PKI), commonly rely on a key management system, and most of the key management tasks are assigned to a trusted third party (TTP) or several distributed TTPs that are based on infrastructure. In contrast, multi-hop cooperative networks are fully decentralized and lack a fixed infrastructure that can act as the TTP. Plus, nodes in such networks have limited memory, computational, and transmission resources. Consequently, the naive solution of storing all keys in every single node for encrypting and decrypting messages is not practical. The two main reasons are the need for a large amount of storage and processing capability to deal with keying operations in large-scale networks, and the fact that storing stationary keys limits the network from updating them or adding new nodes to the network.

Key pre-distribution schemes seem to be a promising solution due to their distributed and lightweight nature. They store just k keys in each node, where $k \ll n$ and n is the number of network nodes. The set of stored keys in each node is referred to as a *keyring*. Once a node encrypts a message with a key, only those nodes with a shared key are capable of decrypting it. Thus, a pair of nodes can communicate directly and securely if they share a common key. To establish a secure connection between two nodes without a shared key, a key-path has to be found. The key-path is an overlay path in which each pair of adjacent nodes have a secure link between them[1], i.e., they share a common key. To exchange messages, the source initially encrypts its message and forwards it to the first hop on the overlay. The message is then routed over the overlay where each intermediate hop, in turn, decrypts the data, encrypts it again with a key shared with the next hop, and forwards it to the next hop toward the destination.

The intermediate decryption-encryption (DE) steps are an obvious security threat in key pre-distribution schemes. To face this threat, several multi-path solutions have been proposed to design a key-exchange process via multiple disjoint paths [2, 3, 4, 5, 6]. The source and the destination nodes after exchanging some keying materials via disjoint paths, agree on a key and use it to encrypt and decrypt their communication. The data communication is then encrypted and decrypted at the endpoints, thus no intermediate decryption-encryption steps remain. Although such multi-path algorithms seem to survive the man-in-the-middle attacks, they are still vulnerable against cooperative attacks such as a cooperative version of the man-in-the-middle attack [6]. The attacker can compromise most of the communication

[1]Note that this secure overlay link may span multiple physical nodes, in reality.

by compromising more nodes. We show in this paper that the multi-path solutions have a strong resistance against node capture attack, but only up until a specific node compromise rate of about 10%. The interesting result is that when the rate of compromised nodes surpasses this threshold rate, it causes significant resistance degradation such that the baseline key pre-distribution schemes show better resistance in comparison with the multi-path solutions.

In this work, we developed BlocKP, a Blockchain-based multi-path key exchange algorithm that improves the resistance of the network against the node capture attack. We propose BlocKP in two versions BlocKP-I and BlocKP-II. We exploit the power of Blockchain in BlocKP-I to improve the resistance using fix-length disjoint paths. The basic idea is to send the keying material via multiple disjoint paths such that the keying material form a block, and this block is recalculated in each hop to form a Blockchain. The destination node, in turn, extracts the keying material from the last block of each Blockchain. The Blockchain helps us to keep the keying materials consistent and guarantees their integrity. To keep the keying material consistent through every Blockchain, we need fix-length disjoint paths. Next, we propose BlocKP-II, a Blockchain-based algorithm which uses erasure coding to improve the resistance of BlocKP-I and relax its assumption of requiring fix-length disjoint paths. We analytically prove the resistance of BlocKP and show how it approaches the perfect resistance, even for a small number of disjoint paths.

The performance of the proposed BlocKP algorithms are evaluated using extensive simulations. First, we evaluate the security strength of BlocKP. Thus, we simulate a mobile ad hoc network with a percentage of compromised nodes. We show that for the network with more than 10% of compromised nodes, BlocKP-I and BloKP-II show about 40% and 60% improvement in the resistance of the network against node capture attack in comparison with the state-of-the-art key exchange solutions. We then simulate a network in ns-2 to evaluate the performance of the proposed algorithm for different number of nodes and different communication loads. In our simulations, we simulate Probabilistic asymmetric key pre-distribution (PAKP), strong Steiner trade (SST), and unital key pre-distribution (UKP) schemes as the baseline key pre-distribution schemes. We augment each of these schemes by a multi-path solution to make their communication secure from end to end. Results show that, in different network setting with different network densities and different network loads, BlocKP decreases the flow completion time up to 20% and improves the network throughput up to 5% in comparison with baseline schemes.

The contribution of this work is to address, for the first time, the common vulnerability problem of the key pre-distribution schemes against the large-scale node capture attack. We propose BlocKP, a Blockchain-based multi-path key exchange solution which *i*) improves the resistance of mobile ad hoc networks against the node capture attack to more than 30% of captured nodes, in its BlocKP-I version; *ii*) reduces the number of required disjoint paths by exploiting the power of erasure coding in its BlocKP-II version. We further mathematically analyze the network's resistance against node capture attacks for several state-of-the-art algorithms and both versions of BlocKP algorithms and compare the analytical results with that of simulations. In addition, we exhaustively evaluate the network performance for different combinations of baseline key pre-distribution schemes, the multi-path solutions, and BlocKP under different network load and densities using the ns-2 network simulator.

3.2 Methods, Assumptions and Procedures

BlocKP is, in essence, a three-phase Blockchain-based key exchange algorithm followed by a secure data transfer— the source node generates a request as a block and sends the block via multiple disjoint paths to form multiple Blockchains (Phase 1). The block is updated in each hop and its consistency is checked until reaching the destination (Phase 2). The destination, in turn, validates the blocks and responds to the request such that the source and the destination become able to extract a pairwise key (Phase 3). The data can be then exchanged securely and efficiently between the source and the destination following the shortest path. Lemma (3.1) shows that the only possible way for an attacker to get access to transferred data is to perform a man-in-the-middle attack, successfully. By calculating the probability of successful man-in-the-middle attacks, we show that BlocKP improves the resistance against node capture attack for any key pre-distribution schemes, regardless of whether its underlying cryptosystem being symmetric or asymmetric. We propose BlocKP in two different versions, BlocKP-I and BlocKP-II. BlocKP-I has a fairly simpler algorithm and is convenient for a better understanding of the general idea of BlocKP. BlocKP-II, on the other hand, optimizes the resistance of BlocKP-I against node capture attack as well as its performance, by exploiting the power of erasure coding.

Lemma 3.1. *Considering the attacker's goal as eavesdropping or altering the transmitted data, using BlocKP, the only possible way to achieving this goal is to perform a man-in-the-middle attack, successfully. Obviously, by altering the transmitted data, we mean changing the data into other meaningful data.*

BlocKP needs a public-private key pair for each node in its key exchange process. Although any asymmetric cryptosystem could be used along with BlocKP, we propose to use elliptic curve cryptography (ECC) for its shorter key length and lower power consumption in comparison with other asymmetric cryptosystems. In our system, each node chooses its own private key and accordingly calculates its corresponding pubic key. Thus, BlocKP does not need any central or trusted party. The use of asymmetric cryptography is limited to signing at most two certificates by the end nodes. Hence, it does not impose a considerable overhead.

BlocKP-I: Blockchain-based key pre-distribution

The general algorithm of BlocKP is as follows. In the first phase, the source node finds multiple disjoint paths toward the destination and builds a block of request. In the second phase, the block follows disjoint paths toward the destination such that in each hop, the block is updated and the hash value of the new block is attached to the block. The majority consensus of the block is checked in each hop. In the third phase, the destination sends a response encrypted by the source node key, following the shortest physical path. The three-phase algorithm makes the pair of source and destination nodes able to calculate a pairwise key. The three phases are then followed by symmetric data encryption using the calculated pairwise key. The encrypted data, in turn, follows the shortest physical path toward the destination. The details of the three phases are as follows.

Phase 1: The source node finds a set $\mathcal{P}$ that contains ρ number of equal-length vertex disjoint paths $p_i, i = 0, 1, \ldots, \rho - 1$. The source node then forms a request packet to get the destination public key. The request packet is considered as the first block of the Blockchain

and represented by B_0. Each block B_i is formed as

$$B_i = (b_i||h(b_i)), \tag{27}$$

where the notation $a||b$ represents the concatenation of a and b,

$$b_i = (i, ID_{src}, ID_{dst}, y_{src}, \mathcal{P}, h(b_{i-1})), \tag{28}$$

y_i is the public key of node i, ID_i is the unique identifier of node i, h is a collision-free hash function, and $h(b_{-1})$ is equal to zero.

Phase 2: For each disjoint path, the source node encrypts the initial block with the shared key of the first intermediate node of that path. The source node then sends each encrypted packet to the corresponding node. Each intermediate node, in turn, decrypts the packet and forms the next block B_1 according to Equation (27). All the intermediate nodes generate a similar B_1 block. Each intermediate node sends B_1 to the next node in its path. It also sends $h(b_1)$ into all next-hop intermediate nodes, for all other disjoint paths. We show in Lemma (3.2), sending the hash value publicly does not increase the chance of the attacker to get access to the encrypted data. Each intermediate node validates the hash value of the received block and checks the majority consensus of the hash value. In case of inconsistency, the intermediate node requests the block with majority consensus from one of the previous level intermediate nodes. In Lemma (3.3), we show that the previous block could be requested and respond publicly. This procedure is continued until the block reaches the destination. At the first-level intermediate nodes, since the block is coming directly from the source node, the majority consensus of the block is not required to be checked.

Lemma 3.2. *Publicly sending the hash value of the block to the next level intermediate nodes, in the second phase of BlocKP algorithm, does not negatively affect the resistance of the network against node capture attack.*

Lemma 3.3. *An intermediate node, in which its received block does not match the majority consensus, can publicly request the block of one of the intermediate nodes with the consensus hash value. The intermediate node, in turn, can also respond publicly. The public request and response in this step do not degrade the resistance of the network against node capture attack.*

Phase 3: The destination validates the source node's public key by checking the majority consensus of the received blocks. It encrypts its own public key by the public key of the source node and sends the encrypted key via a shortest physical path. Now, both the source and the destination have the other party's public key and potentially are able to communicate with each other securely. Due to the computational complexity of the asymmetric encryption, we propose to use symmetric encryption for their communication. The pairwise key can be calculated by both the source and the destination as:

$$K_{sd} = x_{src}.y_{dst} = x_{dst}.y_{src}. \tag{29}$$

At this point, the source and the destination can communicate securely via the shortest physical path. Algorithm (3) represents the pseudocode of BlocKP-I. Considering the length

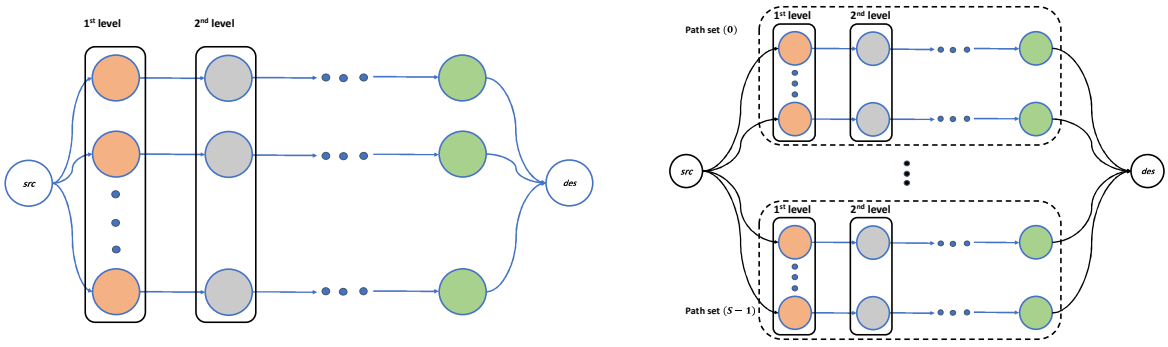
of disjoint paths as l, in Lemma (3.4) we show that the attacker needs to compromise at least the majority of the nodes at one of the $l-1$ intermediate levels to become able to capture the communication. We theoretically show that BlocKP-I significantly increases the resistance of the network against node capture attack in comparison with the state-of-the-art.

Algorithm 2 BlocKP-I($\mathcal{P}$)

Note: $\mathcal{P}$ is a set of ρ equal size disjoint key-paths from the source node toward the destination.
$j = 0$, {j represents the position of the block in the Blockchain, starting from zero at the source node and ending with the key-path length at the destination node.}
while (j <=key-path length) **do**
  if ($j > 0$) **then**
    Receive($e(B_j, p_i(j+1))$),{The intermediate node of the i^{th} path receives an encrypted block from the previous node in of its key-path.}
    B_j= decrypt($e(B_j, p_i(j+1)), x_{p_i(j+1)}$), {Each intermediate node decrypts its corresponding block by its private key. }
    validate($h(b_j)$, {Validate the extracted hash value by comparing it to the one received from the previous level nodes.}
    if NOT-VALIDATED **then**
      request(B_j) , {Block B_j is requested from one of the previous level nodes within the majority. }
    end if
  end if
  if $j <$ key-path length **then**
    $b_j = (j, ID_{src}, ID_{dst}, y_{src}, \mathcal{P}, h(b_{j-1}))$, {Form the block content.}
    $B_j = (b_j || h(b_j))$, {Attach the hash of the block content to the end of the block.}
    for ($i = 0 : \rho - 1$) **do**
      $e(B_j, p_i(j+1)) =$ encrypt$(B_j, p_i(j+1), y_{p_i(j+1)})$, {Encrypt the block with the public key of the next node in path key-path p_i. }
      send($e(B_j, p_i(j+1))$), {Send the encrypted block to the next node of the key-path p_i. }
      if ($j > 0$) **then**
        send($h(b_j)$, ($\forall k,\ k \neq i$) $p_k(j+1)$), {The node sends $h(b_j)$ to the next nodes of all other paths for validation purposes. }
      end if
    end for
  end if
  j=j+1
end while

Lemma 3.4. *In BlocKP-I, the attacker needs to compromise at least the majority of nodes in one intermediate step to be able to forge the public key of the sender and performs the man-in-the-middle attack.*

BlocKP-II: The combination of Blockchain and erasure coding



(a) BlocKP-I (b) BlocKP-II

Figure 4: A schematic view of BlocKP-I versus BlocKP-II.

We propose BlocKP-II to optimize the performance as well as the resistance of BlocKP-I, using erasure coding. Erasure coding is, in essence, a method for forwarding error correction in the case of bit erasure, by expanding a message into a longer message containing redundant data. The main idea of BlocKP-II is to transfer the public key of the source node to several pieces, say S pieces, which are referred to as key shares. Collecting any θ shares reconstructs the original key. However, any $\theta - 1$ pieces or less do not leak any information about the key. Each share of the key is then sent via a set of disjoint paths similar to the first and the second phases of BlocKP-I. The destination, in turn, collects any θ number of shares to reconstruct the key and proceed with the third phase of BlocKP-I. Accordingly, the three phases of BlocKP-II are as the following. Fig. (4) represents a schematic view of both BlocKP-I (Fig. 4a) and BlocKP-II (Fig. 4b).

Phase 1: The source node chooses $\theta-1$ random numbers $a_i, i = 1, 2, \ldots, \theta-1$ and forms the polynomial $P(x)$ as

$$P(x) = y_{src} + \sum_{i=1}^{\theta-1} a_i x^i. \tag{30}$$

$P(0)$ is the public key of the source node and $P(i)$, $i = 1, 2, \ldots, S$ are the key shares. Gathering any θ number of shares leads to reconstructing the public key of the source node.

We use the same number of disjoint paths as BlocKP-I, i.e. ρ. We divide this number into S set of paths $\mathcal{P}_i, i = 0, 1, \ldots, S-1$, each with the ρ_i number of paths such that $\sum_{i=0}^{S-1} \rho_i = \rho$. We name the j^{th} path from i^{th} set $P_{(i,j)}$ where $i = 0, 1, 2, \ldots, S-1$ and $j = 0, 1, 2, \ldots, \rho_i - 1$. The path length in each set has to be constant but different sets may vary in the length of their paths. The source node forms S request packets $B_{(i,0)}$ for $i = 0, 1, \ldots, S-1$. Each block $B_{(i,j)}$ is calculated as

$$B_{(i,j)} = (b_{(i,j)} || h(b_{(i,j)})), \quad \begin{matrix} i = 0, 1, 2, \ldots, S-1, \\ j = 0, 1, \ldots, \rho_i - 1, \end{matrix} \tag{31}$$

where

$$b_{(i,j)} = (j, ID_{src}, ID_{dst}, i, \mathcal{P}_i, sign(y_{src}), h(b_{(i,j-1)})), \tag{32}$$

and $h(b_{(i,-1)}) = 0$. The value of $sign(y_{src})$ is the public key of the source node signed by its

own private key. This value will be used by the destination node for certifying the extracted public key.

Phase 2: In this phase, the source node encrypts the block $B_{(i,0)}$ by the shared key with the first node of each path and sends the encrypted message to the corresponding node. It is worthy to recall that there are S different blocks $B_{(i,0)}, i = 0, 1, \ldots, S-1$, each one has to be sent via its corresponding set of paths $\mathcal{P}_i$. Each intermediate node, in turn, decrypts the block and generates the next block by increasing the block number and changing the previous hash (Equations 31). It then calculates the current hash value and attaches it to the block (Equation 32).

The intermediate node encrypts the new block by the shared key with the next node and sends the encrypted block to the corresponding node. It needs also to send the hash value to the next-hop intermediate nodes of the other paths of its own path set. Each intermediate node checks the majority consensus of the hash values and steps forward by forming the new block. If the current hash value does not match the majority, then the node requests the previous block from one of the previous level intermediate nodes within the majority. This process will be continued until at least θ sets deliver their blocks to the destination node.

Phase 3: In this phase, the destination collects θ number of key pieces to reconstruct the source node public key. Considering the set of key shares as $\mathcal{S}$, the source node public key could be constructed as

$$y_{src} = \sum_{i \in \mathcal{S}} l_i P(i), \tag{33}$$

where l_i is a Lagrange multiplier at the point zero and could be calculated as

$$l_i = \prod_{j \in \mathcal{S}, j \neq i} \frac{-j}{i-j}. \tag{34}$$

The destination checks the validity of the public key by checking the value of $sign(y_{src})$. The destination then encrypts its own public key by the source node's public key and sends the encrypted message directly to the source node via the shortest physical path. By receiving the destination's key, both the source and the destination nodes can calculate the pairwise key, using Equation (29). The data transfer will then follow the shortest physical path after symmetric encryption, using the pairwise key. The pseudocode of BlocKP-II is represented in Algorithm (3). In Lemma (3.5), we show that the attacker needs to compromise at least the majority of one level in at least θ number of path sets to become able to perform man-in-the-middle attacks and forges its own public key. We show that using BlocKP-II improves the resistance of the network against node capture attack in comparison with BlocKP-I to protect the communications against approximately 10% more compromised nodes.

Lemma 3.5. *In BlocKP-II, the attacker needs to compromise at least the majority of one level in θ number of path sets, to be able to forge the public key of the source node.*

3.3 Results and Discussions

We study the security aspects of baseline key pre-distribution schemes, multi-path solutions, and both versions of BlocKP. We analytically calculate the resistance against the node

Algorithm 3 BlocKP-II

$P(x) = y_{src} + \sum_{i=1}^{\theta-1} a_i x^i$, {The source node forms polynomial $P(x)$.}
$\mathcal{P} = findDisjointPaths(src, dst, \rho, S)$, {The source node finds S sets of ρ_i equal size disjoint key-paths toward the destination where $\sum_{i=0}^{S-1} \rho_i = \rho$.}
for $(i = 0 : S - 1)$ **do**
  BlocKP-I($\mathcal{P}_i$), {Perform the BlocKP-I algorithm for each path set $\mathcal{P}_i$.}
end for
$P(x)$ =aggregate($\mathcal{S}$), {The destination collects θ shares of the key to form the set $\mathcal{S}$ and reconstruct the polynomial $P(x)$ by aggregating the shares. }
$y_{src} = P(0)$, {The destination extract the source public key.}

capture attack and support the analytical calculations by simulations. In our simulation scenarios, we consider a network with 1000 nodes in a 1000×1000 m^2 area, moving according to the random waypoint (RWP) mobility model. Further to RWP, we tested random walk, and Levy-walk mobility models to evaluate the effect of movement patterns on the algorithms. We find that the results are fairly similar for all mobility models. Accordingly, we report only the results of the RWP mobility model.

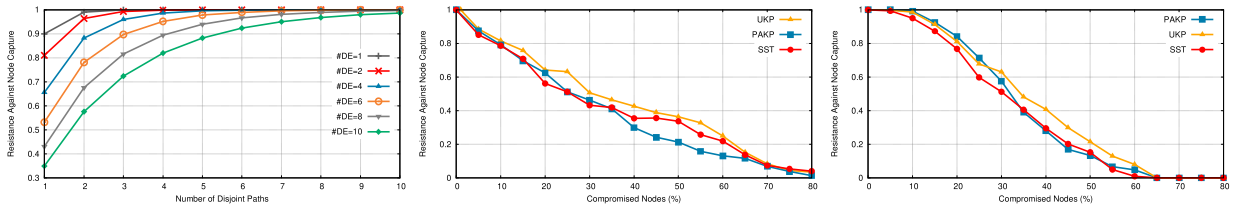
In our simulations, we considered SST, UKP, and PAKP as the baseline key pre-distribution schemes. Our selection for the schemes is such that we cover different categories of key pre-distributions where PAKP is an asymmetric scheme, SST and UKP are symmetric ones. We further augmented each of the baseline schemes with a multi-path algorithm. SST and UKP have been augmented by the algorithm of [3] which is proposed for symmetric key pre-distribution schemes. PAKP has been augmented by the algorithm of [6], designed specifically to work based on PAKP. For multi-path solutions, we considered six disjoint paths where we divide them into three sets of paths for BlocKP-II algorithm. We repeated each simulation scenario to reach 95% confidence for the error to be less than 0.01, according to the Monte Carlo theorem [7]. The reported results are the average of all repeated simulation scenarios.

We calculate a general analytical formula for the resistance against node capture attack of multi-path solutions. Lemma (3.6) shows the formula where ρ and l are the number of disjoint paths and path length, respectively. By letting the number of paths equal to one, the formula results in the resistance of baseline key pre-distribution schemes. In this formula, we assume that the attacker compromises a fraction p of the entire network nodes, uniformly at random. Hence, the probability of each node to be compromised is equal to p.

Lemma 3.6. *The resistance of multi-path solutions against node capture attack could be calculated as*

$$R = 1 - (1 - (1 - p)^{l-1})^{\rho}. \tag{35}$$

Fig. (5a) shows the results of Equation (35) for a network with 10% of its nodes being compromised. We can conclude from this figure that the multi-path solutions do not need a large number of disjoint paths to reach a high resistance against node capture attack. Since most of the multi-path solutions send only a few control packets through the multiple-paths and then use the shortest paths for data transmission, they do not impose serious performance degradation. We show that not only is the overhead of such solutions negligible



(a) (b) (c)

Figure 5: A resistance comparison against node capture attack: (a) Analytical results for multi-path solutions with 10% of the nodes being captured; (b) Simulation results for baseline schemes; (c) Simulation results for multi-path solutions.

but also they improve the overall performance by eliminating the path stretch. Fig. (5b and 5c) show the simulation results of baseline schemes and multi-path solutions, respectively, for different fractions of compromised nodes. Considering that the average disjoint path length in our setting is 2.28, 2.42, and 2.58 for PAKP, UKP, and SST, respectively, the simulation results certainly validate our analytical formula. It further shows that although the resistance against node capture attack in multi-path solutions is much higher than that of simple key pre-distribution schemes, after a specific threshold of captured nodes, the resistance of multi-path solutions is reduced and becomes worse than that of baseline schemes. However, baseline key pre-distribution schemes represent fairly linear degradation in their resistance by the increment in the number of captured nodes. Lemma (3.7) and (3.8) analytically calculate the resistance of the network against captured nodes for both BlocKP-I and BlocKP-II, respectively.

Lemma 3.7. *The resistance of BlocKP-I algorithm against node capture attack is equal to*

$$R(\rho, l) = \left(1 - \sum_{i=*\frac{\rho}{2}+1}^{\rho} \binom{\rho}{i} (p^i)(1-p)^{(\rho-i)}\right)^{(l-1)}. \tag{36}$$

We represent this parameter by $R(\rho, l)$ as it depends on the number of vertex-disjoint paths (ρ) and the number of DE steps in each path $(l-1)$.

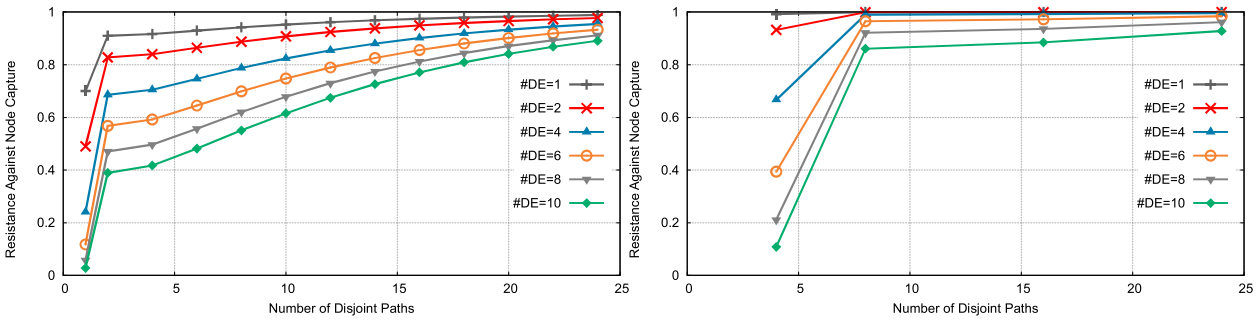
Lemma 3.8. *The resistance of BlocKP-II against node capture attack can be calculated as*

$$R = 1 - \left(\sum_{i=\theta}^{S} \binom{S}{i} (1 - R(\rho_i, l_i))^i (R(\rho_i, l_i))^{S-i}\right), \tag{37}$$

where $R(\rho_i, l_i)$ could be calculated by Equation 36. Considering $\theta = S$, this formula will be reduced to

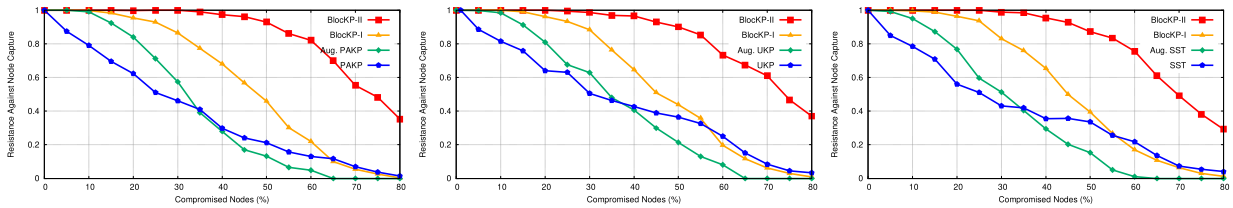
$$R = 1 - \prod_{i=1}^{S} (1 - R(\rho_i, l_i)).$$

Fig. (6) shows the analytical results of resistance against node capture attack in a network with 30% of its nodes captured for BlocKP-I, i.e. Lemma (3.7), and BlocKP-II, i.e. Lemma



(a) BlocKP-I (b) BlocKP-II

Figure 6: A comparison of the resistance against 30% of the nodes in a network being captured (analytical results).



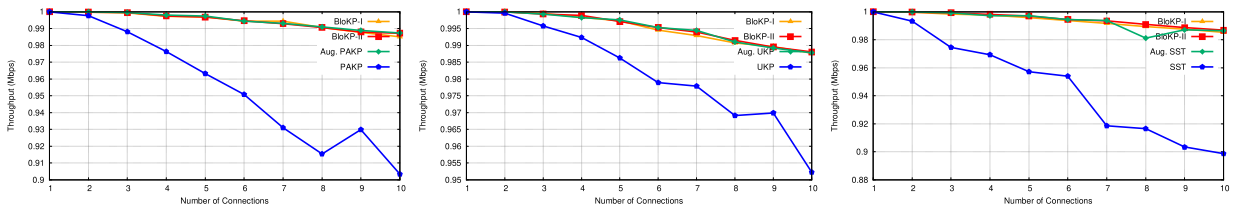
(a) PAKP (b) UKP (c) SST

Figure 7: A general comparison of resistance against node capture attack among baseline schemes, multi-path solutions, and both versions of BlocKP algorithms (simulation results).

(3.8). To validate the analytical results and to have a general overview of how powerful the BlocKP algorithm is, Fig. (7) shows the simulation results of the resistance against node capture attack for all the combinations of baseline schemes, multi-path solutions, BlocKP-I, and BlocKP-II. Considering that the number of disjoint paths in these simulations are six, and the average number of intermediate D-E steps are 2.28, 2.42, and 2.58 for PAKP, UKP, and SST, respectively, the results show a certain validation to our analytical calculations, i.e. results of Fig. (6).

W further make a comprehensive performance comparison between the different combinations of baseline key pre-distribution algorithms, augmented ones, and both versions of BlocKP. We show that besides the security improvement of BlocKP, its performance is also much higher than that of baseline key pre-distribution schemes and fairly comparable to that of multi-path solutions of the state-of-the-art. We consider throughput, flow completion time (FCT), key-exchange delay, routing traffic, and key exchange overhead as the performance metrics.

Simulation Setting: We use the ns-2 network simulator [8] to simulate a mobile ad hoc network in a 300×300 m^2 area. To investigate the scalability of different algorithms, we simulate networks starting with 100 nodes and increasing to 200 in increments of 10. In each simulation instance, we choose five pairs of source and destination nodes, uniformly at random. For each pair, a random start time is chosen uniformly in the interval $[0 \quad 60] sec.$ Once the set of source-destination pairs is chosen, the same set is used for the simulation of



(a) PAKP (b) UKP (c) SST

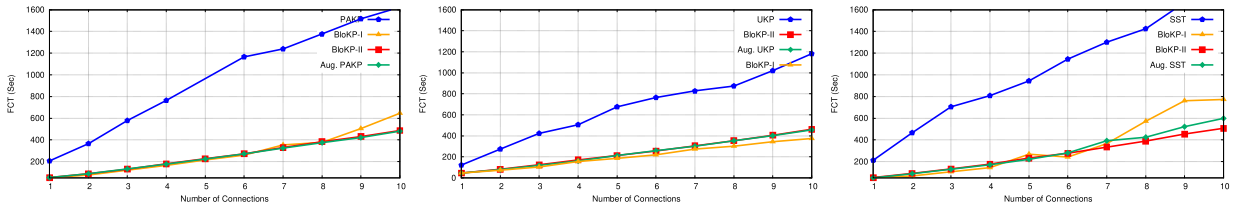
Figure 8: A comparison of the throughput for different key pre-distribution schemes.

all algorithms. The source node starts to send a file of 5 MB for its corresponding destination using TCP Tahoe on a channel with 1 Mbps bandwidth. To investigate the performance of the network under different loads, in a 100 node setting, we simulate the network for a number of connections ranging from 1 to 10. We find that increasing the number of nodes does not have a major effect on the pattern of the results. Hence, we reported only the results of the network with a fix number of nodes and different numbers of connections. We run each simulation instance for all the combinations of baseline key pre-distribution schemes, including PAKP, SST, and UKP, their augmented version, and their versions supported by both BlocKP-I and BlocKP-II. We use AODV as the underlay routing protocol. The nodes are assumed to be mobile following RWP, random walk (RW), and Levy walk (LW) mobility models. We find that the results show the same pattern for all the mentioned models. Thus we represent only the results of the RWP model in this paper. A distance model with a communication range of 100 m is considered for all nodes.

For the key pre-distribution process, we consider PAKP, SST, and UKP schemes. In SST, the probability of storing a shared key between any pair of nodes does not exceed 0.25, which leads to longer key path length. However, the attacker needs to compromise more nodes to be able to access all keys. In 2-UKP, the probability of storing a shared key is much higher, which leads to shorter key paths, but the attacker can get access to all distributed keys by compromising a small numbers of nodes. Each node stores ten keys, and six disjoint paths are considered for multi-path solutions. For BlocKP-II we divide the set of paths into three subsets, each with two disjoint paths.

Simulation Result: We now investigate the network throughput, flow completion time, and control traffic as the performance metrics. We further compare the delay and control traffic of the key exchange process for those algorithms that include such a step. We repeat each simulation scenario 20 times, and the presented results are the average of all simulated scenarios. We measure the throughput as the ratio of successful packet delivery over the bandwidth. Fig. (8) shows this metric for different schemes and their augmented algorithms. Since the simple baseline schemes suffer from path stretch problem, they show lower throughput in comparison with the multi-path solutions. In multi-path solutions, the data transfer process follows the shortest path between the source and the destination, after a key-exchange process.

The next measured metric is flow completion time. We measure this metric as the time between the first packet sent by the source node and the last packet of the 5 MB file received by the destination. This time includes the time required for the key-exchange process in multi-path algorithms. Fig. (9) shows the results of FCT for different baseline schemes and



(a) PAKP (b) UKP (c) SST

Figure 9: A comparison of the flow completion time for different key pre-distribution schemes.

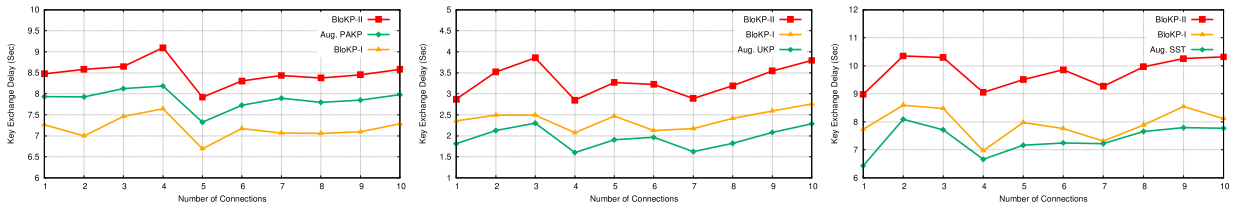
their multi-path extensions. Since the increment in the number of connections leads to more congestion, the FCT increases by the increment in the number of concurrent connections. Again, the baseline key pre-distribution schemes have longer FCT due to the longer key paths.

Fig. (10) shows only the time consumed for the key-exchange process for different multi-path solutions. Aug. UKP and aug. SST algorithms have just one phase in their key exchange process. These algorithms send different key pieces via disjoint paths and then immediately start sending the data packets. The destination becomes able to decrypt data packets after receiving all key pieces. Accordingly, we calculate the time required for the key exchange process as the time starting from the transfer of the first key exchange packet by the source node until receiving the last piece of the key by the destination. This parameter for aug. PAKP, since it includes one more phase, is calculated as the time between sending the first key exchange packet and receiving the destination key by the source node. While aug. UKP and aug. SST show the shortest key exchange process, BlocKP-I generally outperforms BlocKP-II for all algorithms. However, BloKP-I needs a set of same length disjoint paths, which in some cases might not be available. Thus, we calculated the key exchange failure ratio as the ratio of unavailability of enough number of fixed length disjoint paths. Table (2) shows the failure ratio for all multi-path solutions. This table shows that even for the SST scheme, which suffers from low key sharing probability, BlocKP-II failure rate approaches to zero. However, BlocKP-I suffers from a higher failure rate in comparison with other solutions. Since in the PAKP scheme, the key paths have to be disjoint only in the overlay, a lower failure rate could be seen.

Table 2: Failure ratio of the multi-path algorithms.

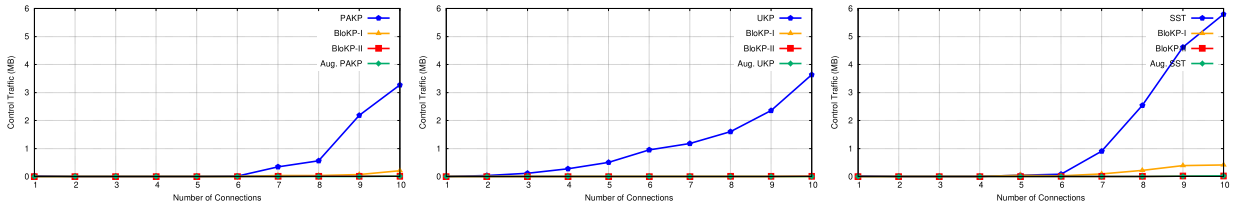
Scheme	PAKP			SST			UKP		
Algorithm	Aug.	BloKP-I	BlocKP-II	Aug.	BloKP-I	BlocKP-II	Aug.	BloKP-I	BlocKP-II
Failure Rate	0.0	0.117	0.003	0.115	0.240	0.003	0.0	0.168	0.0

It is well known that in dynamic networks, the amount of control traffic is one of the main performance evaluation metrics. This metric includes routing traffic, key exchange traffic, and so on. We measured the average control traffic per each connection for different numbers of connections. Fig. (11) shows the results. By the increment in the number of



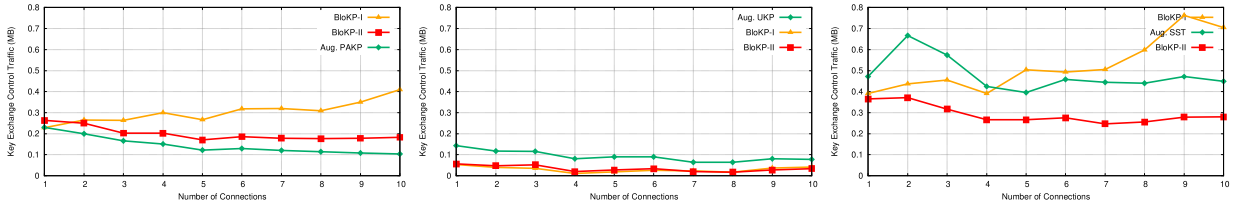
(a) PAKP (b) UKP (c) SST

Figure 10: A comparison of the key exchange delay for different key pre-distribution schemes.



(a) PAKP (b) UKP (c) SST

Figure 11: A comparison of the end-to-end control traffic for different key pre-distribution schemes.



(a) PAKP (b) UKP (c) SST

Figure 12: A comparison of the key exchange control traffic for different key pre-distribution schemes.

concurrent connections, we see an obvious increment in the control traffic of the baseline key pre-distribution schemes. The main reason is the longer path length in comparison with multi-path solutions. BlocKP-I shows slightly higher traffic volume due to its overall longer key-path length.

We further measured the control traffic of the key exchange process for multi-path solutions. Fig. (12) shows the results. The high key sharing probability in the UKP scheme, which results in shorter paths along with the lightweight nature of the Aug. UKP key exchange process, leads this scheme to have relatively less control traffic in its key exchange process. The low key sharing probability in SST, in turn, leads this scheme to experience more control traffic. Generally, BlokcKP-II generates a lower volume of control traffic in comparison with BlocKP-I, due to its lower key-path length.

3.4 Conclusions

The intermediate D-E problem caused by key pre-distribution schemes is a dire security threat where the multi-path algorithms were known as effective solutions. In this project, we analytically showed that although the multi-path solutions are effective, they are vulnerable against a high ratio of node capture. We supported and validated our analytical results with simulations. To face this deficiency, we proposed the BlocKP-I algorithm based on Blockchain and improved its idea by exploiting the power of erasure coding to propose BlocKP-II. We analytically showed that BlocKP could resist the high ratio of node capture, even for more than 50% of entire network nodes. We further performed exhaustive simulations to show the performance of BlocKP in both versions. We showed that BlocKP improves the network performance in comparison with the baseline key pre-distribution schemes, and it has a comparable performance with the literature multi-path algorithms. Since energy consumption is crucial in multi-hop networks, we suggest a comprehensive power evaluation for the key pre-distribution algorithms, as future work. As another future direction, we recommend implementing the BlocKP algorithm in the real world. Designing different attack scenarios to evaluate the sensitivity of the key pre-distribution-based algorithm is also of paramount importance.

4 Outcomes

1. Mohammed Gharib, Fatemeh Afghah, and Elizabeth Bentley. "OPAR: Optimized Predictive and Adaptive Routing for Cooperative UAV Networks". In Proceedings of the **IEEE INFOCOM 2021 WORKSHOPS**, 14th International Workshop on Wireless Sensor, Robot and UAV Networks, Vancouver, Canada, May 2021.

2. M. Gharib, S. Nandadapu, F. Afghah, "An Exhaustive Study of Using Commercial LTE Network for UAV Communication in Rural Areas", In Proceedings of the **IEEE ICC Workshop on Integrating UAVs into 5G and Beyond**, 2021.

3. Mohammed Gharib, Fatemeh Afghah, and Elizabeth Bentley, "LB-OPAR: Load Balanced Optimized Predictive and Adaptive Routing for Cooperative UAV Networks," submitted to **Ad hoc Networks journal** , 2021.

4. Mohammed Gharib, Ali Owfi, Fatemeh Afghah, and Elizabeth Bentley, "BlocKP: Key Pre-Distribution Based Secure Data Transfer" submitted to **IEEE IoT Journal** , 2021.

References

[1] M. Gharib, M. Minaei, M. Golkari, and A. Movaghar, "Expert key selection impact on the manets' performance using probabilistic key management algorithm," in *Proceedings of the 6th International Conference on Security of Information and Networks*, ser. SIN '13. New York, NY, USA: ACM, 2013, pp. 347–351. [Online]. Available: http://doi.acm.org/10.1145/2523514.2523556

[2] P. Papadimitratos and Z. J. Haas, "Secure message transmission in mobile ad hoc networks," *Ad Hoc Networks*, vol. 1, no. 1, pp. 193 – 209, 2003.

[3] Hui Ling and T. Znati, "End-to-end pairwise key establishment using multi-path in wireless sensor network," in *GLOBECOM '05. IEEE Global Telecommunications Conference, 2005.*, vol. 3, Nov 2005, pp. 5 pp.–.

[4] G. Li, H. Ling, and T. Znati, "Path key establishment using multiple secured paths in wireless sensor networks," in *Proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology*, ser. CoNEXT '05. New York, NY, USA: ACM, 2005, pp. 43–49.

[5] J.-P. Sheu and J.-C. Cheng, "Pair-wise path key establishment in wireless sensor networks," *Computer Communications*, vol. 30, no. 11, pp. 2365 – 2374, 2007, special issue on security on wireless ad hoc and sensor networks.

[6] M. Gharib, A. Owfi, and S. Ghorbani, "Kpsec: Secure end-to-end communications for multi-hop wireless networks," 2019.

[7] M. H. Kalos and P. A. Whitlock, *Monte Carlo Methods*. WILEY-VCH, 2008.

[8] (2020, Sep.). [Online]. Available: http://www.isi.edu/nsnam/ns/

List of Symbols, Abbreviations, and Acronyms

Ad hoc On-Demand Distance Vector (AODV)

Boolean Linear programming (BLP)

Breadth-first Search (BFS)

Decryption-encryption (DE)

Destination-Sequenced Distance-Vector Routing (DSDV)

Dynamic Source Routing (DSR)

Elliptic Curve Cryptography (ECC)

Flow Completion Time (FCT)

Gauss-Markov (G-M)

Levy walk (LW)

Linear programming (LP)

Load Balancing optimized predictive adaptive routing algorithm (LB-OPAR)

Optimized Link State Routing (OLSR)

Optimized Predictive Adaptive Routing Algorithm (OPAR)

Probabilistic asymmetric keypre-distribution (PAKP)

Public Key Infrastructure (PKI)

Random Walk (RW)

Random Waypoint (RWP)

Software-defined Networking (SDN)

Strong Steiner Trade (SST)

Transmission Control Protocol (TCP)

Trusted Third Party (TTP)

Unital Key Pre-distribution (UKP)

Unmanned aerial vehicles (UAVs)

Unmanned Terrestrial vehicles (UTVs)