Security of Unmanned Aerial Vehicles: Dynamic state estimation under cyber-physical attacks

Leonard Petnga[§], Huan Xu[†]

***Abstract*— The goal of this work is to detect faults and cyber-physical attacks on unmanned aerial vehicles (UAVs) using dynamic state estimation to determine the nature of such vulnerabilities. We develop and introduce a distributed UAV architecture to characterize attacks and their propagation. Central to the distributed control architecture is a supervisory controller that relies on active sensing and actioning of controllable contactors to collect the needed information for system state estimation using an adaptive algorithm. Uncertainty is reduced through adaptive reconfiguration of the system based on correlation of measurements from sensor readings and prior information on system state. We show that an adaptive greedy strategy algorithm guarantees theoretical worst-case performance for various cyber-physical attack scenarios. This is demonstrated by the results of prototype implementations and simulation on a subgraph of the UAV in the case of a gain control attack.**

***Index Terms*— Unmanned Aerial Vehicle (UAV), Data integrity, Cyber-physical attacks, Cybersecurity.**

I. Introduction and motivation

With the ever-growing use of unmanned aerial vehicles (UAVs) in both military and civilian domains, safety and the ability to withstand cybersecurity threats is becoming a foremost problem. UAVs, as well as other cyber-physical systems (e.g., power grid, transportation, etc.) need to be designed in a manner that accounts for such attacks. Because decisions are based on the state of the cyber-physical system, knowledge gained from data collection and processing are critical to the success of the UAV mission. Furthermore, the strategic value of information enclosed in military UAVs make them valuable targets to espionage, thefts, manipulation and attacks of all kinds. Security forces around the world worry of scenarios in which UAVs are hijacked by remote attacker(s) and are used against soft and sensitive targets. In recent years, several incidents involving cyber attacks on UAVs are illustrations that this scenario could become reality soon. In 2009, insurgents in Iraq successfully intercepted and distributed US military drone video feeds in their network [5]. In September 2011, a "keylogging" virus infected a US military UAV fleet at Creech Air Force Base in Nevada [20], followed three months later by the loss of an RQ-170 Sentinel to Iranian forces [14].

These incidents are a stark reminder of the challenges the research community and industry face in securing UAVs. When the system is under cyber attack, effective estimation of its state is critical for any successful recovery action or maneuver. This task proves especially challenging if sensors are compromised during the attack. Because the state of the system is heavily dependent on sensor measurements, the problem of state estimation from sensor readings is crucial to the safety of the entire system and its mission.

Estimation of UAV state (e.g., position, orientation and velocity) is an active research topic. A large body of work exists on the development of estimation techniques and methods for effective navigation in challenging environments including GPS-denied environments [3], [18], [24], cluttered surroundings and neighborhoods [11] and windy conditions [15]. These approaches rely on localization algorithms based on variations and extensions of filters such as the Gaussian Particle Filter [3] and the Extended Kalman Filter (EKF) [12], [1] to fuse sensor measurements and maintain accurate state estimates. Researchers have also investigated and experimented feedback control-based algorithms [2]. However, Langelaan [11] points out that navigation is only one of many critical flight tasks. Aviation and communications are equally critical to successful flights and missions. Thus, navigation-based estimation approaches of the state of the UAV are inefficient in capturing the full grasp of the status of the system, especially when it is under attacks such as the ones identified above. In such circumstances, it is important to have an estimation of the system state that accounts for the internal state of the components in order to: (1) establish/infer the type and extend of the attack and, (2) provide lead(s) on possible pathway(s) to effective recovery. This extension of the scope of the system state parameters exacerbates the already difficult challenge of optimal sensor placement in UAVs, especially small ones [23], [19], [8].

Our overall research goal is to provide theoretical guarantees on the ability to estimate security threats to cyber-physical systems in general, and UAVs in particular. The main objective of this work is to model and develop a simulation in which an adaptive submodularity-based mathematical framework can be used to assess cyber threats (both internal and external) to the system. Thus, we aim to obtain state estimates with a limited number of sensors by utilizing software-based dynamic estimation strategies. We are particularly interested in detecting and localizing faults in the system as result of attacks and use that information to establish and characterize cyber-physical threats. To that aim, we build from lessons learned in fault diagnostics [17] and previous applications in aircraft electric control system [13] to discretize continuous flow of data and signal as well as health statuses of components in the system before

[§]Department of Civil and Environmental Engineering, University of Maryland, College Park, MD 20742, correspondence: lpetnga@umd.edu

[†]Department of Aerospace Engineering, University of Maryland, College Park, MD 20742

performing state estimation. Although the application area targets UAVs, the results of this work can be applied to multiple cyber-physical systems domains.

We limit the scope of this work to the cruising operational mode, when the command and control of the UAV system is performed by the autopilot. We investigate and model the standalone UAV system as well as security threats to which it is exposed. The state estimation mathematical formulation uses a property of set functions called adaptive submodularity [4]. Finally, simulation will be conducted to verify the theoretical results.

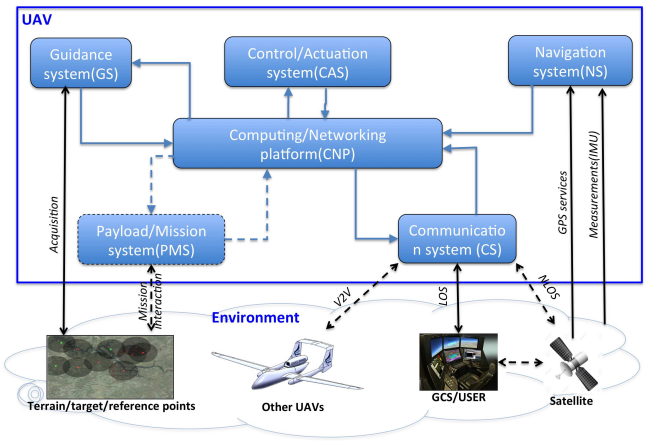


Fig. 1. High level architecture of a UAV. The system is made of six interconnected subsystems plus the payload (mission dependent). Four of them exchange data and information with a complex mix of systems in the environment including other UAVs.

II. System and Attack Modeling

A. UAV system modeling for threat analysis

For the purpose of this work, we adopt a cyber-physical perspective of the UAV system. This view is consistent with the tight coupling and bidirectional interactions between cyber (command, control, communication) and physical (sensors, actuators) components in such systems.

1) High level architecture: For the sake of simplification, we use the UAV airframe as system boundary as shown in Figure 1. Thus, we model the system as an integrated set of five modules/subsystems: (1) Guidance System-GS, (2) Control/Actuation system-CAS , (3) Navigation System-NS, (4) Communication system-CS and (5) Computing and network platform-CNP. While the boundaries of these modules are not always clearly defined in all UAV systems, they are fairly common to the vast majority of UAV models used in dynamics studies [10], [7] and cyber attack analyses [9], [6]. The presence and configuration of the payload/mission system module are dependent on the application and use of the UAV. Therefore, they are kept outside of the scope of this work. The environment of the system is complex and diversified with elements including satellite, ground control system and other UAVs.

The inner structure of the modules, shown in the single line diagram in Figure 2, shows the decomposition into cyber (in white) and physical (in yellow) components

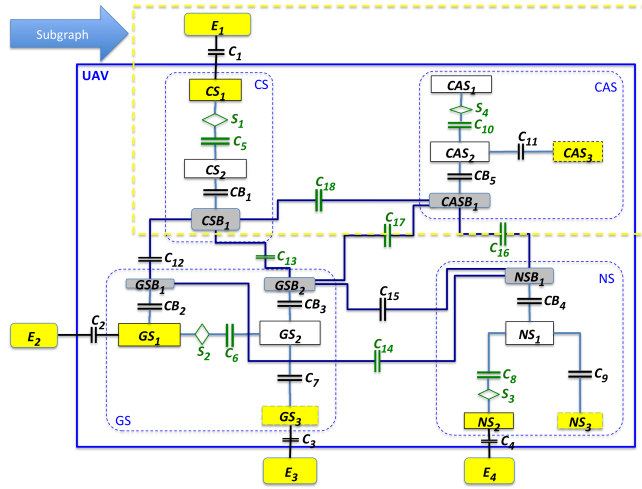


Fig. 2. Simplified single line dataflow diagram of a UAV. The graph shows the decomposition of the main subsystems into components and their connections. Cyber components appear in white while physical components are in yellow. Buses (e.g. CSB_1, GSB_1) serve as interfaces between modules while sensors (S_i with $i \in (1,4)$) measure and report the state of the components and contactors (C_j with $j \in (1,17)$) control the flow of signal/data in the system.

and the interactions between them within and beyond the given module. Entities E_i, $i \in (1,4)$, NS_j, $j \in (1,3)$, CAS_k, $k \in (1,3)$, GS_l, $l \in (1,3)$, CS_m, $m \in (1,2)$ respectively make up the set E, NS, CAS, GS, and CS of environment, navigation, control and actuation, guidance and communication system components. The environment is composed of the ground system transceiver E_1, other UAV ADS-B transceivers E_2, the terrain/target/reference points E_3 and the satellite E_4. All external components (E_i) are modeled as physical entities while individual subsystems are hybrid. The navigation system (NS) is composed of physical components such as the GPS receiver (NS_2) and set of navigation instruments (IMU, magnetometer, etc.) grouped as NS_3 and the filter responsible for NS sensors fusion (NS_1) which is a cyber entity. Inside the control and actuation (CAS) module, the main system controller (CAS_2) is located between a gain controller (CAS_1) and the set of actuators (CAS_3). Similarly, the guidance system is composed of an ADS-B transceiver (GS_1) connected to a path planner (GS_2) which reads in guidance sensors feeds (GS_3), (e.g., vision and radar). Finally, a simplified view of the communication system shows the transceiver (CS_1) for data exchange with ground systems and communication protocols (CS_2) as main components.

2) Sensors, contactors and bus positioning: The connections between the components in Figure 2 illustrate possible data exchange paths for the proper operation of the system. We assume the existence (not shown in the figure) of a supervisory controller that tracks the state of the system. To that aim, it needs a set of sensors placed at strategic locations along the dataflow paths. For the purpose of this research, they are depicted as follows.

Buses serve as the interface between the module or subsystem and the rest of the system. A bus here (e.g.: $CSB_1, CASB_1$) is a connection point for data flows (in and out) for the subsystem components to which it is attached.

Sensors capture the state of (physical) components interacting directly with active entities within the UAV environment. The sensor (e.g., S_1, S_2, S_3) is positioned at the port communicating with the UAV cyber component. A sensor can also capture the state of cyber components that could be the source of cyber infections/attacks (e.g.: S_4). Sensors for internal feedback control are ignored (not watched by the supervisory controller). Sensor measurements are normalized to feed the supervisory controller with the state of the component as either (0) no data/signal flow, (1) within the acceptable range, or (2) outside the acceptable range. These measurements are contingent to the sensor being itself healthy or unhealthy at the time the reading occurs.

Contactors are "dataflow breakers" that can be opened or closed and are mounted on connections between components. Contactors that are controlled (in green) are accessible and actionable by the supervisor. They are either associated with sensors for control of (signal/data) flow to/from controlled components (e. g.: C_5, C_6, C_8, C_{10}) or flows between critical subsystem interconnections - 1 per interconnection when many exist (e.g.: $C_{13}, C_{14}, C_{16}, C_{17}, C_{18}$). Uncontrolled contactors (e.g.: C_7, C_9, C_{12}) are not accessible/controlled by the supervisory controller. In the context of this study, an open uncontrolled contactor can be interpreted as the break of the communication link between two components as the result of an attack or some fault in the system.

B. Simplified threat modeling

Elements and connections along the dataflow are susceptible to attacks. In order to effectively characterize known and future attacks on the UAV, we first create a threat modeling framework that makes use of the CPS prospective introduced in section II-A while accounting for the types of attacks and their propagation mechanisms.

1) Attack modeling and categorization: Researchers have been actively developing threat modeling techniques and approaches [21], [28] and investigating cyber attacks [25] on CPS such as UAVs and automobiles. However, cross domain or cyber-physical threats are equally important, given the potential catastrophic consequences of such attacks [22], [16]. Yampolskiy et al. introduce a taxonomy for description of cross-domain attacks on CPS [26] and a language describing attacks on such systems [27]. We build from these work to generate a simplified classification of attacks on our UAV as modeled in section II-A. Attacks are categorized into four classes, independently of their origin as shown in table I. An *Influenced element* is the type of the object (source) of the attack while a *Victim element* refers to the type of the object that has been influenced by the attack (effect).

2) Attack propagation: In Table I, both *Influenced element* and *Victim element* can be different, thus, the issue of propagation of attack between components arises. Therefore, we ought to be able to understand and describe mechanisms through which cyber and cyber-physical attacks propagate in our system. To that aim, we consider *physical interdependencies* between components which nicely espouse the functional perspective of our modeling approach introduced in section II-A. It is also consistent with the system dataflow structure as per Figure 2. Because of their reliance or dependence on implementation details, other possible interdependencies (geographic, cyber or logical) were left aside in order to preserve the result of this work from the side effects of early design commitments. In the case of a gain scheduling attack, a possible scenario could be as follows. For different flight modes (take off, landing, and cruising), the autopilot needs different gains for flight control depending on the UAV state (mass, altitude, speed, flaps down, etc). For effective control of the vehicle, the autopilot will require that appropriate gains been loaded for each flight phase. An attacker can successfully access the UAV control system (most likely on the ground, before the flight) and override/alterate the pre-computed gains stored in the on-board autopilot. With the gain controller (CAS_1) compromised, the controller (CAS_2) will also be affected as it relies on corrupted gains to track the guidance path and stabilize the aircraft for the corresponding flight mode(s). It now sends inappropriate commands to actuators (CAS_3) thus, affecting the system dynamics with possible multiple safety consequences. This scenario illustrates a situation in which an attack on a cyber component propagates to other components through the interdependency connections and ultimately leads to physical consequences. Thus, this is clearly a C2P attack (i.e. of category II). Because of its open architecture, complex and numerous physical interdependencies, there is almost no end to the list of possible attacks and threats to UAVs. Therefore, there might be multiple branchings to the propagation tree of a given attack. An analysis of vulnerabilities such as fuzzing attack, GPS spoofing or digital update rate attack [9] are some illustrations. This requires a more detailed classification system which is beyond the scope of this work.

III. Mathematical problem formulation and State estimation strategy

A. General problem description

We consider a graph $G = (\mathcal{N}, \mathcal{E})$ representing the UAV architecture shown in the data flow diagram in Figure 2. The various system components - communication (CS_i with $i \in (1,2)$), navigation (NS_j with $j \in (1,3)$), guidance (GS_k with $k \in (1,3)$), control and actuation (CAS_l with $l \in (1,3)$) - along with their buses and sensors (S_i, $i \in (1,4)$) are elements in the set of nodes $\mathcal{N}$ of G. Similarly, contactors (C_i, with $i \in (1,17)$) and physical connectors between components are elements of the set of edges $\mathcal{E}$. Contactors $\mathcal{C} \subseteq \mathcal{E}$ can either be *open* or *closed*. Entities in the set of communication $\mathcal{M}$, navigation $\mathcal{I}$, guidance $\mathcal{U}$, control and actuation $\mathcal{A}$ are uncontrollable. We call $\mathcal{P}$ the set of components in the graph i.e. $\mathcal{P} = \mathcal{M} \cup \mathcal{I} \cup \mathcal{U} \cup \mathcal{A}$, with $\mathcal{P} \subset G$. The state of components in $\mathcal{P}$ can either be *unhealthy* (i.e., the component is online but outputting data/signal

Category	Description	Influenced Element	Victim Element	Example attacks
I	Cyber over Cyber(C2C)	Cyber	Cyber	Buffer overflow, DoS, man in the middle, etc.
II	Cyber over Physical(C2P)	Cyber	Physical	Stuxnet, Gain control
III	Physical over Cyber(P2C)	Physical	Cyber	Side-channel attack, Sensor jamming
IV	Physical over Physical(P2P)	Physical	Physical	Physical tempering, Propagation of C2P, etc.

TABLE I

ATTACK CLASSIFICATION AND CHARACTERIZATION

outside the acceptable operational range), *healthy* (i.e., the component is online and outputting data/signal within the acceptable operational range), or *offline* (i.e., the component is not online and/or there is no data/signal flow). These states are read by the supervisory controller as introduced in section II-A.2. Nodes associated to pure system level sensing functionality such as E_3, E_4 or NS_2, NS_3, GS_3 have only outgoing edges and no incoming edges. All other edges in the graph are bidirectional.

Sensors $\mathcal{S} \subseteq \mathcal{N}$ are special nodes in the graph G. Their readings are dependent on the status of their components (in $\mathcal{P}$) as well as the one of the contactor located on the link between the selected pair of components. For a sensor reading to be considered, there needs to be an "active" path between the two nodes of the graph corresponding to the components involved. In other words, the following three conditions should be satisfied: (1) there exists a simple path in G that connects the two nodes, (2) no component along that path is offline and, (3) all contactors along the path are closed. Under these conditions, the possible readings of a sensor $s \in \mathcal{S}$ can be one of the following values (i) *unacceptable value* if there is an active path between s and some $p \in \mathcal{P}$ (not offline) which is unhealthy; (ii) *acceptable value* if, for all $p_i, p_j \in \mathcal{P}$, $i \neq j$ that have an active path to s, each pair (p_i, p_j) along such paths is made of components that are healthy; or (iii) *no signal/dataflow* if there is no active path between s and any component $p \in \mathcal{P}$.

In this framework, the supervisory controller acts as an embedded controller siting on top of the distributing sensing and control architecture. It is the ultimate responsible for the system dynamic state estimation and it is able to detect faults and abnormal system behaviors resulting from (cyber-physical) attacks. To that aim, the supervisory controller controls a subset $\mathcal{C} \setminus \mathcal{C}'$ of contactors (i.e., C_5, C_6, C_8 and C_{10} in Figure 2).

We express the state x of the system as a valuation of the state of individual components $p \in \mathcal{P}$ and uncontrollable contactors $e \in \mathcal{C}' \subseteq \mathcal{C}$. Given the limited amount of sensors and the presence of uncontrollable contactors in the system, the state x of the system is unknown. Thus, we model it as a random variable X whose values are possible system states mapped to sensor readings. We would like to develop a fault detection adaptive estimation strategy the supervisory controller will use to figure out the discrete state of the system for various cyber-physical attack configurations. The adaptive estimation strategy is performed by the controller which "acts" on the system by opening and closing controllable contactors, then, reads sensors measurements.

B. Mathematical formulation

We base the mathematical formulation from Golovin and Krause [4]. For further details, we also refer the reader to [13]. What follows is a summary of the mathematical frame of the dynamic state estimation problem.

The set of all system states is defined as Ω while the state X of the system is modeled as a random variable. A probability measure $\mathbb{P}[x]$ is built on the set Ω based on data components and reliability levels. The initial state is the unknown state $x_0 \in \Omega$ and it is assumed unchanged during the estimation process. We also assume independence between faults in the system. The rationale behind these assumptions is that the timescale of the estimation process, by design, is expected to be much smaller than the failure rates of the components and the timescales of other controllers in the system.

1) Formulating the optimal estimation strategy: There exists a set $\mathcal{V}$ of actions v, in the controllable subset of contactors, and a set $\mathcal{Y}$ of measurements y that can be observed. For an action $v \in \mathcal{V}$, $y = \mu(v, x) \in (Y)$ is the unique outcome of performing action v if the system is in the state x. The actions $\{v_0, ..., v_t\}$ performed, and outcomes $\{y_0, ..., y_t\}$ observed up until step t, are represented by the partial realization $\psi_t = \{(v_i, y_i)\}_{i \in \{0,...,t\}}$. At each step t, the probability measure $\mathbb{P}[x]$ can be updated by conditioning it on ψ_t to obtain $\mathbb{P}[x \mid \psi_t]$.

An effective estimation process should adaptively eliminate "invalid" states to get to the actual state x_0. We define $D(y, v)$ with $y = \mu(v, x_0)$, as the set of states $x \in \Omega$ that are indistinguishable from x_0 under the action v. Similarly, we introduce S_t as the set of states that produce the same set of outcomes $\{\mu(v_0, x_0), \ldots, \mu(v_t, x_0)\}$ as x_0 under the same set of actions $\{v_0, \ldots, v_t\}$.

To represent the uncertainty in the state estimate, we define an objective function $f : 2^{\mathcal{V} \times \mathcal{Y}} \times \Omega \rightarrow \mathbb{R}_+$ that maps the set of actions $A \subseteq \mathcal{V}$ under state x_0 to reward $f(A, x_0)$. A strategy π is a function from partial realizations to actions such that $\pi(\psi_t)$ is the action v_{t+1} taken by π when observing ψ_t. The fault detection controller is assigned a budget $k \ll |\mathcal{V}|$, indicating the number of steps within which the estimation process should terminate, starting from the initial (unknown) state x_0. Then, for $i = 1, \ldots, k$, a sequence of decision, measurement and update operations is performed as part of the estimation process.

With the goal of reducing the uncertainty of X represented by the probability distribution $\mathbb{P}[x]$ through performing k

actions, the following reward function is considered:

$$f(v_{0:k}, x_0) = -\mathbb{P}[S_k] = -\sum_{x \in S_k} \mathbb{P}[x]. \quad (1)$$

Therefore, maximizing f is equivalent to removing as much probability mass (uncertainty) as possible from Ω in k steps. When $\mathbb{P}$ is uniform, f becomes proportional to the size of S_k. In such situation, maximizing f is equivalent to minimizing the number of indistinguishable states. Thus, our ultimate goal is to devise the estimation strategy π^* that delivers the "best expected estimate" for the state as follows.

$$\pi^* \in \arg\max_{\pi} \mathbb{E}[f(\tilde{\mathcal{V}}(\pi, X), X)], \quad (2)$$

subject to $|\tilde{\mathcal{V}}(\pi, x)| \leqslant k$ for all x, and with expectation taken with respect to $\mathbb{P}[x]$. Here, $\tilde{\mathcal{V}}(\pi, x) \subseteq \mathcal{V}$ is the set of all actions performed under the strategy π; the state of the system being x.

2) Greedy strategy and guarantee of its worst case execution performance : The supervisory controller needs a fault detection strategy able to plan ahead for k steps. However, in equation (2), the complexity of the optimal strategy scales up exponentially with k. The greedy strategy in [13] solves this issue while maximizing the one-step expected uncertainty reduction problem. Recent results in adaptive submodularity are exploited to provide theoretical guarantees for its worst-case execution performance [4]. In brief, the strategy picks, at each step, the action maximizing the expected one-step gain in uncertainty reduction. At a given step t, it uses the available information ψ_t to compute the probability measure $\mathbb{P}[x \mid \psi_t]$ on the set S_t, using the Bayesian update:

$$\mathbb{P}[x \mid \psi_t] = \begin{cases} \frac{\mathbb{P}[x]}{\mathbb{P}[\psi_t]} & \forall\, x \in S_t \\ 0 & elsewhere \end{cases} \quad (3)$$

At each step t, the strategy consists of choosing the next action v_{t+1} that maximizes the gain in uncertainty reduction. The value of the function f in equation (1) is used as measure of uncertainty and the benefit is expressed in terms of its change as a new action v is chosen. Thus, in other to realize the estimation goal, we choose to maximize the mean benefit at each step under the expectation taken with respect to the updated probability measure $\mathbb{P}[x \mid \psi_t]$. This leads to the greedy strategy :

$$v_{t+1} \in \arg\max_{v \in \mathcal{V}} \mathbb{E}[f(v_{0:t} \cup \{v\}, X) - f(v_{0:t}, X) \mid \psi_t]. \quad (4)$$

Given the fact that greedy strategies are known to deliver bad performance arbitrarily, recent results from adaptive submodularity research come in handy to provide lower bounds to their performance. With respect to the issue of guaranteeing worse case performance for the greedy strategy, a brief overview of those results can be found in the appendix of [13]. As for the conditions to be satisfied by the reward function f used by the greedy strategy in equation (4), it has to be adaptive and submodular; both properties demonstrated to be satisfied by f. Therefore, the following result follows.

Theorem 1: For any true state $x_0 \in \Omega$, the uncertainty reduction achieved in k steps by the greedy strategy given in Algorithm 1 is no worse than $(1 - 1/e)$ of what can be achieved in k steps by any other strategy, including the best possible strategy [4].

IV. Implementation and Prototype Simulation

In this section, we illustrate the implementation of the dynamic state estimator employing the adaptive submodularity-based greedy algorithm on a simplified UAV topology such as the one introduced in section II-A, in the context of cyber-physical attacks.

A. Estimation process

Algorithm 1 Adaptive greedy strategy

Input: Probability measure $\mathbb{P}[x]$ on Ω, number of actions to perform k. The system is in the state $x_0 \in \Omega$, fixed and unknown, and the controlled contactors are in some configuration v_0.
Output: Partial realization ψ_k and the set S_k of compatible states after k actions are taken based on the strategy π_{greedy}

```
Take the measurement y_0 = μ(v_0, x_0).
ψ_0 = {(v_0, y_0)}
for t ∈ {1, ..., k} do
    v_t = π_greedy(ψ_{t-1})
    Perform action v_t
    Take the measurement y_t = μ(v_t, x_0)
    ψ_t = ψ_{t-1} ∪ {v_t, y_t}
    S_t = S_{t-1} ∩ D(y_t, v_t)
    Compute P[x | ψ_t] (Bayesian update)
end for
return (ψ_k, S_k, P[x | ψ_k])
```

Algorithm 1 provides a summary of the estimation process. In other to improve computation efficiency, some steps can and will be computed offline. As an illustration, we consider the inverse mapping from sensor measurements to appropriate component states. In the absence of a close form expression for the sets $D(y, v)$ of states for the action-measurement pairs (v, y) (where $v \in \mathcal{V}$ is an action and $y \in \mathcal{Y}$ is the resulting measurement), its computation will require a straighforward but thorough and repeated search for paths on the graph G. Thus, the pairs (v, y) are computed offline, stored in a database, then accessed on the fly at run time. Also, the estimation framework supports the encoding and integration of assumptions about the components, the system architecture and, most importantly the cyber-physical attacks. Given the focus of this work on the latter aspect, we will assume that an attack has happened on an existing system i.e. a UAV. Assumptions reduce the initial of size of the state set Ω by eliminating infeasible states.

In order to evaluate the performance of the greedy strategy in estimating the state of the UAV while under attack, we systematically test the dynamic estimator in Algorithm 1 on increasingly extensive portions of the dataflow diagram in Figure 2. The selection of the size and boundaries of the subgraph to consider will remain consistent with the findings in section II-B.2 on attack propagations. For both simplification and experimentation purpose, we assume a

uniform probability distribution over Ω. Thus, the size of the feasible set is reduced to the reward function f (see Eq. (1)).

Increasingly stringent constraints on weight, wingspans (volume) and cost - especially on small UAVs - force designers and systems engineers to limit the number of sensors onboard. In such situations, it is virtually impossible for the supervisory controller to guaranty a determinate, uncertainty-free assessment of the state of the system. We check the performance of the greedy strategy implemented by the supervisory controller against a brute force strategy, which exhaustively and systematically tries every single authorized action $v \in \mathcal{V}$. Despite its obvious advantage of gathering and providing more information than the greedy strategy, the brute force strategy is impractical as the size of $\mathcal{V}$ i.e. $|\mathcal{V}|$ can be very big. However, it can serve as performance benchmark to the greedy strategy. For the rest of this paper, we perform the test methodology described in Algorithm 2.

Algorithm 2 Simplified test methodology

Input: Initial configuration of the controlled contactors $v_0 \in \mathcal{V}$
1: **for** $x_0 \in \Omega$ **do**
2: Set the graph in the state x_0
3: Put the controlled contactors in the configuration v
4: Run the strategy tested (Greedy or brute force strategy)
5: Record the computation time and the value of f at the end for statistics.
6: **end for**

B. Simulations on a UAV subgraph

1) Simulation set up: We consider the subgraph comprising the UAV communication system (CS) and control and actuation system (CAS) as delimited by the yellow dashed box in Figure 2. The subgraph is made of 8 nodes (components) including, 1 external component (E_1), 2 communication components (CS_1, CS_2) , 3 control and actuation components (CAS_1, CAS_2, CAS_3), 2 buses ($CSB_1, CASB_1$) and 7 contactors ($C_1, C_5, C_{11}, C_{10}, C_{18}, CB_1, CB_5$). Among the contactors, 3 (C_5, C_{10}, C_{18}) are controlled by the supervisory controller. All sensors are assumed to be healthy (state = "h"), i.e., they properly capture and represent the state of the component/flow they are sensing/measuring. We use the gain control attack described in section II-B.2 as the baseline for our experiments. For the sake of simplification, we will assume that the ground system transceiver (E_1) is healthy. Therefore, the configurations in table II are considered and encoded during simulation as assumptions on the subgraph of interest whenever needed for this attack (category II).

2) Simulation results: Simulation platform. Our system specifications are as follows: $Intel^{®}$ $Core^{TM}$ i5-1600M 64 bits CPU 2.5 GHz, 8.00 Gb RAM. The code is written in $Python$. Both brute force and greedy strategies were run on the very same platform for the configurations identified in table II . On this particular example, there are three controlled contactors. Therefore, the brute force strategy performs $|\mathcal{V}| = 2^3 = 8$ actions.

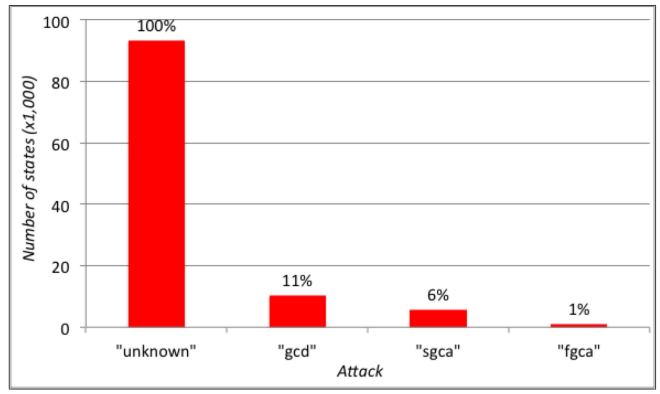


Fig. 3. Histogram of the size of the search space. The latter reduces drastically as assumptions on faults pile up. Also, the performance of both the brute force and greedy strategies is the same when the greedy strategy is executed with a horizon length of $k = 6$.

Size of the state-space. Figure 3 shows the histogram of the size of the search space for the configurations identified in table II. It appears that the size of the search space, taking into account the assumptions on faults, reduces drastically as assumptions pile up. It drops 99%, from the initial 93,000 states under "none" configuration to less than 1,200 states under "fgca" configuration. Also, the performance of both strategies is the same when the greedy strategy is executed with a horizon length of $k = 6$. Specifically, for a given attack, the value of the objective function f at the end of the 6 steps using the greedy strategy is the same as after the brute force strategy with 8 steps.

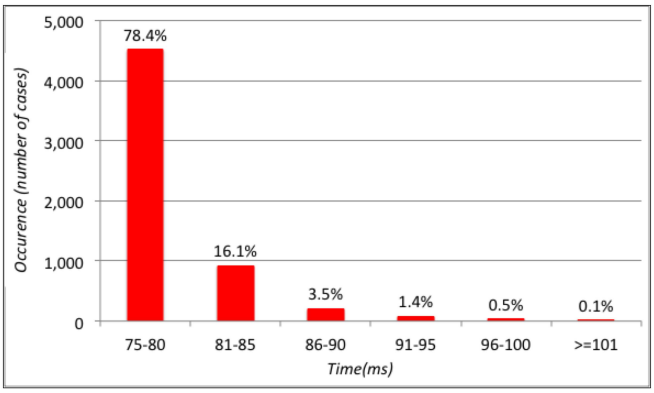
Average execution time. For this experiment, we select the "sgca" and "fgca" configurations and summarize in Figure 4(a) and 4(b) the average execution time for the greedy strategy. Under suspected gain control attack assumption, in 80% of cases, it takes less than 80 ms to compute the next best action to perform using the greedy strategy against 73% of cases and just 6 ms for the full gain control attack configuration. Overall, the graph shows that the greedy strategy is very fast. However, the offline computation takes significantly more time under the "sgca" configuration (7min 30s) than under the "fgca" configuration (8s). This discrepancy can be explained by the relative low size of the search space under "fgca" configuration (1%) as opposed to the "sgca" configuration (6%) with respect to the full space as shown in Figure 3.

Final value of the reward function f. Figure 5 shows the performance comparison between greedy and brute force strategies. It appears that the greedy strategy performs as well as our benchmark i.e. the brute force strategy. For a point at coordinates (a, b) in the graphic, in $a\%$ of the cases, there are b or fewer states that can't be distinguished after the given strategy runs its normal course of action. The number of indistinguishable states under a specific attack assumption remains unchanged. However, it increases as assumptions are progressively relaxed. Overall, the greedy strategy matches the performance of the brute force strategy. After the $k = 6$ steps, the search state is reduced to only 4.2% of its initial

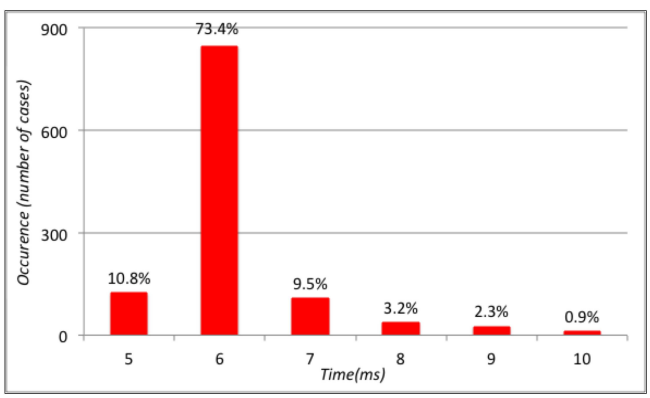
Configuration	Description	Assumptions encoded
"none" or "unknown"	No specific attack assumed	None i.e. no assumption on infected component. The full state space is used
"gcd"	Gain controller down	Only the gain controller (CAS_1) is infected
"sgca"	Suspected gain controller attack	The gain controller (CAS_1) <u>and</u> the controller (CAS_2) <u>or</u> the actuator (CAS_3) are infected.
"fgca"	Full gain controller attack	<u>All</u> CAS components i.e. the gain controller (CAS_1, the controller (CAS_2) and the actuator (CAS_3) are infected.

TABLE II

ATTACK ASSUMPTIONS AND CONFIGURATIONS ENCODING FOR SIMULATION.



(a) Execution time-"sgca".



(b) Execution time-"fgca".

Fig. 4. Average execution time for various attack configurations. Under suspected gain controller attack or "sgca" configuration, it takes less than 80 ms in 80% of cases to compute the next best action to perform using the greedy strategy against just 6 ms in 73% of cases for the full gain control attack or "fgca" configuration.

size by the greedy strategy in all configurations except in the first one (i.e. "none") where it performs even better with a reduction to just 1.4% of the initial size.

V. DISCUSSION

The complexity of the dataflow diagram can grow exponentially with the number of components and connections between them. Solving the dynamic state estimation problem on the resulting graph for such complex architectures can be very computationally and time expensive. Thus, we need strategies to simplify the graph by reducing its size. Whenever it is possible, clustering certain components together into *metacomponents* is a possible solution. However, this

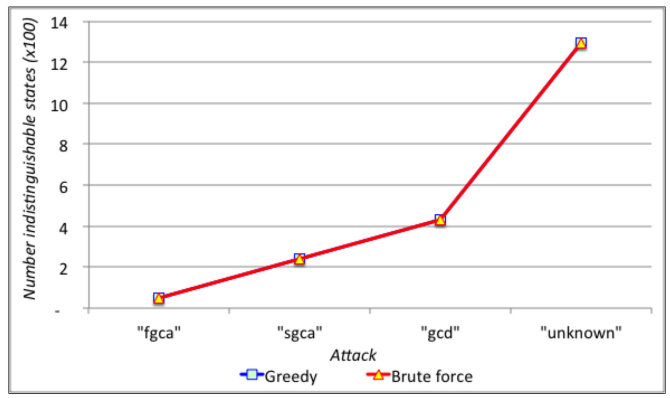


Fig. 5. Performance comparison between greedy and brute force strategies. The greedy strategy performs as well as the benchmark i.e. the brute force strategy. The number of indistinguishable states under a specific attack assumption remains unchanged. However, it increases as assumptions are progressively relaxed.

approach should preserve the quality of the state estimation process. Specifically, the reduction of the complexity of the diagram can be achieved by abstracting away connections between two uncontrolled components that do not have any sensor on their internal connecting port. In this case, some of the individual states of the components may become indistinguishable from what can be measured with the available sensors. The global behavior of these metacomponents will be similar to the one of a single basic component. Therefore, estimating the states of individual components will require first, the estimation of the state of the metacomponent then, its mapping to possible states of corresponding basic components. Thus, one needs to make the necessary adjustments when running state estimation algorithms on the reduced graph in order to guarantee lossless abstraction in the transformations.

Simulations have shown that the greedy strategy performs well with respect to all the metrics considered and in comparison to the brute force strategy. However, real-world problems are bigger, more complex and require extensive computation resources (memory and space on the disk). Model reduction mechanisms like the one just described might not be good enough to keep the computation load in check. As an illustration, we have tried to implement the full graph in Figure 2 in order to support the analysis of other attacks such as the GPS spoofing and the digital update rate attacks. It didn't work out well as we quickly ran out of memory, even under the most stringent set of

assumptions. Thus, we should consider coupling lossless abstraction mechanisms to sensors repositioning strategy in order to successfully scale up this approach to larger architectures.

Also, we need to stress the trade-off existing between the number of sensors available and the complexity of the estimation process. More sensors allows a better estimation by reducing the uncertainty on the state, but also generates a growing complexity resulting in the increase of the computation time. Moreover, more sensors leads to more weight and less (mission) payload for the UAV.

VI. Conclusion and future work

In this work, we investigate cyber-physical attacks on UAVs and the use of a greedy strategy for dynamic state estimation to characterize such vulnerabilities. A simplified and generic UAV architecture accounting for internal signals/dataflows is introduced and used to support both attacks characterization and propagation and algorithm implementation and testing. Sensors positioning for effective system state estimation is considered. The algorithm guarantees theoretical worst-case performance for various cyber-physical attack scenarios. Moreover, it performs as well as the brute force strategy as demonstrated by the results of prototype implementations and simulation on a subgraph of the UAV in the case of gain control attacks. The estimation process generates either the set of all observable states or, when applicable, the unique feasible state of the system.

As preliminary results have shown, there is a need to go beyond the reliance on a simple taxonomy to better understand and tackle the complexity of the propagation mechanisms of attacks and handle them in attack models. Thus, future work should look at candidate formal, UML-based languages to capture those complex interactions and attack models as a whole in a systematic way. There is also a need to formulate and integrate safety requirements in the estimation process. One illustrative issue is the ability to gain insight into the algorithm performance bounds under time-constrained changes of actions. We believe the actual support of our framework for assumptions is a way forward. The ability to eliminate unsafe actions from the initial set can be encoded in a similar way. Finally, the number and location of sensors in the system topology are architecture and application-dependent. Integrating sensors placement studies and analysis in the system design loop will provide an optimal solution to this problem thus, ultimately ameliorate the performance of the system state estimation process.

Acknowledgement

This work was supported in part by MITRE Corporation.

References

[1] J. D. Barton. Fundamentals of small unmanned aircraft flight. *Technical Digest*, 31(2):132–149, 2012.

[2] J. D. Barton. Suas code: Uav state estimation examples. 2012.

[3] A. Bry, A. Bachrach, and N. Roy. State estimation for aggressive flight in gps-denied environments using onboard sensing. IEEE International Conference on Robotics and Automation (ICRA), 2012.

[4] Krause A. Golovin, D. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, pages 427–486, 2011.

[5] S. Gorman, Y. Dreazen, and A. Cole. Insurgents hack u.s. drones. http://www.wsj.com/articles/SB126102247889095011, Retrieved 25 January 2016, 2009.

[6] K. Hartmann and C. Steup. The vulnerability of uavs to cyber attacks - an approach to the risk assessment. 5th International Conference on Cyber Conflict, 2013.

[7] J. B. Hstmark. Modelling simulation and control of fixed-wing uav: Cyberswan. MS Thesis, Engineering and Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway, 2007.

[8] N. Kandasamy, F. A. Aloul, and T. J. Koo. Sensor selection and placement for failure diagnosis in networked aerial robots. IEEE International Conference on Robotics and Automation Orlando, Florida, USA, 2006.

[9] A. Kim, B. Wampler, J. Goppert, and I. Hwang. Cyber attack vulnerabilities analysis for unmanned aerial vehicles. Infotech aerospace, Garden Grove, CA, USA, 2012.

[10] R. Klenke. Development of a novel, two-processor architecture for a small uav autopilot system. Virginia Commonwealth University, School Of Engineering, Electrical Engineering, Richmond, VA, 2006.

[11] J. Langelaan. State estimation for autonomous flight in cluttered environments. PhD Dissertation, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, 2006.

[12] T. Magnusson. State estimation of uav using extended kalman filter. MS Thesis, Department of Electrical Engineering, Linkopings University, Sweeden, 2013.

[13] Q. Maillet, H. Xu, N. Ozay, and R. M. Murray. Dynamic state estimation in distributed aircraft electric control systems via adaptive submodularity. IIEEE Conference on Decision and Control, Florence, Italy, 2013.

[14] S. Peterson and P. Faramarzi. Exclusive: Iran hijacked u.s. drone, says iranian engineer. csmonitor.com, Retrieved 25 January 2016, 2011.

[15] D. Poorman. State estimation for autopilot control of small unmanned aerial vehicles in windy conditions. MS Thesis, Department of Aerospace Engineering Sciences, University of Colorado, CO, USA, 2014.

[16] G. Raz. Hacking drones and the dangers it presents. http://www.npr.org/2012/07/08/156459939/hacking-drones-and-the-dangers-it-presents, Retrieved 25 January 2016, 2012.

[17] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D.C. Teneketzis. Failure diagnosis using discrete-event models. *Control Systems Technology, IEEE Transactions on*, 4(2):105–124, 1996.

[18] S. Saripalli. State estimation for aggressive flight in gps-denied environments using onboard sensing. International Conference on Environmental and Agriculture Engineering (ICEAE 2009), 2009.

[19] H. Semke, K. J. J. Lemler, and M. Thapa. An experimental modal channel reduction procedure using a pareto chart. *Topics in Modal Analysis II*, 8:101–110, 2014.

[20] N. Shachtman. Computer virus hits u.s. drone fleet. http://www.wired.com/2011/10/virus-hits-drone-fleet/, Retrieved 25 January 2016, 2011.

[21] C. Vestlund. Threat analysis on vehicle computer systems. MS Thesis, Department of Computer and Information Science, Linkoping University, Linkoping, Sweeden, 2010.

[22] J. Villasenor. Cyber-physical attacks and drone strikes: The next homeland security threat. http://www.brookings.edu/research/papers/2011/07/05-drones-villasenor, Retrieved 25 January 2016, 2011.

[23] M. Vitus and C. Tomlin. Sensor placement for improved robotic navigation. Robotics: Science and Systems VI, 2010.

[24] D. Wu, E. Johnson, M. Kaess, F. Dellaert, and G. Chowdhary. Autonomous flight in gps-denied environments using monocular vision and inertial sensors. American Institute of Aeronautics and Astronautics, 2013.

[25] M. Yampolskiy, P. Horvath, X. D. Koutsoukos, Y. Xue, and J. Sztipanovits. Systematic analysis of cyber-attacks on cps evaluating applicability of dfd-based approach. pages 55 – 62. 5th International Symposium on Resilient Control Systems (ISRCS), Salt Lake city, UT, USA, 2012.

[26] M. Yampolskiy, P. Horvath, X. D. Koutsoukos, Y. Xue, and J. Sztipanovits. Taxonomy for description of cross-domain attacks on cps. HiCoNS '13 Proceedings of the 2nd ACM international conference on High confidence networked systems, 2013.

[27] M. Yampolskiy, P. Horvath, X. D. Koutsoukos, Y. Xue, and J. Sztipanovits. A language for describing attacks on cyber-physical systems. *International Journal of Critical Infrastructure Protection*, 8:40–52, 2015.

[28] J. Zalewski, S. Drager, W. McKeever, and A. J. Kornecki. Threat modeling for security assessment in cyber-physical systems. CSIIRW'2012, ACM 978-1-4503-1687-3/12/10, Oak Ridge, Tenn., USA, 2013.