Special Issue Article

Journal of Defense Modeling and Simulation: Applications, Methodology, Technology
1–17
© The Author(s) 2016
DOI: 10.1177/1548512916628335
dms.sagepub.com

Securing unmanned autonomous systems from cyber threats

Bharat B Madan[1], Manoj Banik[1], and Doina Bein[2]

Abstract
Unmanned systems, with and without a human-in-the loop, are being deployed in a range of military and civilian applications spanning air, ground, sea-surface and undersea environments. Large investments, particularly in robotics, electronic miniaturization, sensors, network communication, information technology and artificial intelligence are likely to further accelerate this trend. The operation of unmanned systems, and of applications that use these systems, are heavily dependent on cyber systems that are used to collect, store, process and communicate data, making data a critical resource. At the same time, undesirable elements of our society and adversarial states have also realized the high value of this resource. While enormous efforts have been made to secure data and cyber systems, lack of rigorous threat modeling and risk analysis can lead to more specific, rather than generic, security solutions relevant to the cyber system to be protected. This scenario has created an urgent need to develop a holistic process for protecting data and cyber systems. This paper deals with the development of different pieces of this process. We first identify the security requirements of unmanned autonomous systems, and follow this up with modeling how attacks achieve their objectives. We argue that a large number of threats that can materialize as attacks and the costs of managing these attacks in cost effective ways require ranking threats using cyber threat modeling and cyber risk analysis techniques. The last segment of the paper describes a structured approach to mitigate high-risk threats.

Keywords
Unmanned systems, autonomous systems, unmanned aerial vehicle, unmanned ground vehicle, unmanned underwater vehicle, cyber security, attack modeling, threat modeling, risk analysis, intrusion mitigation, intrusion tolerance

1. Introduction

Unmanned autonomous systems (UASs) can be classified into two broad classes as follows.

- Remotely controlled or remotely piloted systems (henceforth referred to simply as *unmanned systems* [USs]) that operate directly or indirectly under the control of a remote human operator. The remote operator sends control commands typically over communication links (e.g. cable, wireless or satellite) to control the system or the vehicle behavior. The received commands are executed by the unmanned vehicle's embedded control systems to control its behavior. In addition, embedded sensors measure the vehicle's internal state and external environmental variables. The sensed information is used internally by the vehicle and also sent to the operator and data analysts.[1]
- Self-controlled systems (henceforth referred to as *autonomous systems* [ASs]) work by perceiving their internal state and external environment using

[1]Department of Modeling, Simulation & Visualization Engineering, Old Dominion University, USA
[2]Department of Computer Science, California State University–Fullerton, USA

Corresponding author:
Bharat B Madan, Department of Modeling, Simulation & Visualization Engineering, Old Dominion University, 1322 ECSB, 4700 Elkhorn Avenue, Norfolk, VA 23529, USA.
Email: bmadan@odu.edu

sensors to compute and implement intelligent control decisions autonomously by processing the multi-sensor data on its embedded computer systems. Typically, such an AS also needs to include the functionality of an unmanned vehicle to allow it be remotely controlled and managed in exceptional situations. In fact, autonomously computed operational decisions dealing with certain exceptional situations (e.g. firing an on board weapon) may have to be authorized by its human supervisor(s), that is, human-in-the-loop mode of operation.

Despite minor differences, as we will see in the sequel, USs and ASs have similar security requirements. A typical AS requires much more sophisticated software for its functioning compared to an US. Thus, an AS's software security requirements are likely to be more demanding as compared to that of a typical US. Conversely, since an US needs to constantly interact with its operator over electronic communication channels, its communication security requirements are likely to be more extensive as compared to that of an AS. Notwithstanding such differences, within the scope of this paper, we will refer to both these class of systems as *unmanned vehicles* (UVs).

UVs are being deployed for civilian and military application on the ground, in the air, sea-surface and underwater at a rising pace.[2] At the present time, civilian applications of UVs are confined to a small set of applications, mostly of an exploratory nature, for example, merchandise delivery, monitoring of power lines, monitoring offshore oil and gas pipelines and future modes of transportation. Use of UVs in military applications is much more extensive and growing at an exponential rate to the extent that over the last few years, United States military doctrine has become highly UV-centric as compared to other major militaries. UVs are being deployed extensively on the ground, in the air and at sea or underwater (unmanned underwater vehicles (UUVs)). A recent Department of Defense report[3,4] highlights the need for accelerating the deployment of UVs for civilian and military applications. DARPA's Urban Grand Challenge competition[5] and Google Car[6] reflect the thinking that unmanned ground vehicles (UGVs) will have an important role in future ground transportation and military ground operations, particularly related to explosive ordinance disposal (EOD).[7] Military UUVs and unmanned surface vehicles (USVs) are being used in naval (e.g. mine countermeasures[8], underwater surveillance, surface combat operations[9]) and civilian applications (e.g. offshore oil and gas well exploration and maintenance and seafloor mapping).

The capture of the stealthy unmanned surveillance aircraft RQ-170 Sentinel in 2011 highlights the challenges likely to be faced by UASs operating in hostile environments. Based on public reports,[10] one can only speculate on how this capture was carried out. However, authoritative reports indicate that this RQ-170 was captured while performing surveillance of Iran's nuclear weapons development program. Since photographs showed the airframe to be intact, it seems likely that it was not shot down. Other plausible reasons for its capture include the following:

- malfunctioning or jamming of command and control (C2) links leading to crash landing;
- hijacking the vehicle by taking over its C2 links;
- spoofing of global positioning system (GPS) transmitters.

These possibilities highlight the need for adding capabilities to UASs that would enable an UAS to infer either autonomously or with the help of available command, control, communications, computers, intelligence, surveillance and reconnaissance (C4ISR) assets whether it is being subjected to unauthorized and malicious actions that can lead its technology and surveillance data being stolen and/or its mission being compromised.

Another known instance of attack on USs involved the compromise of one of the main ground control systems (GCSs) at a United States Air Force base.[11] The attack involved the injection of malware whose malicious payload contained key logger software for stealing passwords and other personal information. While the exact source of this malware could not be established, its source may have been an insider, a worm transported via networks into the GCS, a Trojan hidden inside another software package or an update installed into the GCS systems.

From the cyber security perspective, UASs are also a good example of the so-called Cyber Physical Systems (CPSs). Defending CPSs from cyber threats is much more challenging than for conventional cyber systems. These challenges stem from the fact that a CPS, such as an UAS, often requires real-time control and may have safety as an important system requirement. In addition, an UAS is made up of multiple independent systems, that is, a System-of-Systems (SoS), and many of the individual systems are often commercial-off-the-shelf (COTS) subsystems and components sourced from multiple and diverse vendors. Requirements for securing CPSs as listed below have been identified in a workshop report by the Cyber Security Research Alliance.[12]

- Rapid adoption of COTS technology and protocols can leave vulnerabilities in firmware and software incorporated in a CPS.
- Long lifespans of CPSs combined with evolution of security mechanisms have to be considered in defining security requirements.

- CPSs are exposed to exploits via communications systems connected to the Internet used for providing remote access to maintenance and operational personnel anywhere in the world;
- Integration of CPSs across facilities and multiple autonomous companies across the globe increases the attack surface.
- Many CPSs use inherently insecure protocols (e.g. MODBUS) that increase the risks of connected operations due to poor authentication practices.
- Use of legacy CPSs and industrial control systems in critical infrastructure may not have enough memory or processing power for integrating security protocols and strong authentication practices.
- Dependence on legacy components with little or lack of support is an additional source of vulnerabilities in CPSs.
- Lack of personnel training in security continues to be an issue.
- CPSs may have been designed and implemented by vendors or suppliers that no longer exist. Heavy use of legacy components with little or no support is yet another source of vulnerabilities in CPSs.
- Management changes could cause reduced focus on security.
- Requirements for uninterrupted operations can prevent changes, fixes or upgrades.

The remainder of this paper is organized into four sections. Section 2 identifies different security attributes that need to be protected. Section 3 discusses various types of threats faced by an UAS, along with a model of how an attacker exploits underlying vulnerabilities in an UAS to create a threat to one or more of these security attributes. Section 4 discusses the process of threat modeling and risk analysis that can be used to prioritize different threats and describes how the Common Vulnerability Scoring System (CVSS) tool implements this process; Section 5 provides the concluding remarks.

2. Security attributes

A system, such as an UAS, that employs cyber subsystems for its control and management is defined in terms of the security of data or information during the in-storage, in-process and in-transit phases. (Henceforth, we will use *data* and *information* interchangeably, despite the fact that *information* is derived by processing *data* so that data can be interpreted suitably in the context of an application.) During all three phases, security of data can be defined in terms of five security attributes:

Confidentiality: ability of a system that assures that it is capable of preventing access to data by any unauthorized entity;
Integrity: ability of a system that assures that it is capable of preventing unauthorized alterations of data by any entity;
Availability: ability of a system that assures that data and system resources are made available within specified time to all authorized entities;
Authentication: ability of a system to verify the identity of a subject; and
Non-repudiation: ability of a system to ensure that actions performed by an entity cannot be refuted afterwards.

However, traditionally *Confidentiality, Integrity* and *Availability* security attributes are frequently used to measure the security performance of cyber systems.[13,14]

An UAS consists of multiple systems that are integrated together to work as a single SoS. Figure 1 shows the conceptual architecture of an unmanned aerial vehicle (UAV). The architecture of an UGV and UUV is conceptually similar, with minor additions and deletions of some subsystems. For example, ADS-B has no relevance to UGVs and UUVs. An UUV may be equipped with sonars and side scan sonars in place of radars and synthetic aperture radars, which may be present in UAVs. Similarly, an UGV may have a ground penetrating radar instead of the airborne radar on an UAV. Notwithstanding these differences, all three types of UASs face similar generic cyber threats, which are annotated in red in Figure 1.

Cyber attackers exploit vulnerabilities in software, security policies and communication technologies to gain abilities to perform unauthorized actions to compromise confidentiality, integrity and/or availability of data while data is in various states, as summarized in Table 1. It should also be realized that designing and launching a well-crafted attack on a well-defended UAS is a complex task. The so-called divide-and-conquer strategy is a proven strategy for dealing with all types of complex tasks. In the context of cyber-attacks, this strategy takes the form of structuring a complex attack as a sequence of inter-related *atomic attacks* in the form of a tree. Each such atomic attack is designed to compromise a particular security attribute, that is, confidentiality or integrity or availability, such that an atomic attack allows the attacker to move closer to its final objective.

Schneier[15] introduced the term *Attack Tree* (AT) to formalize such an attack strategy. In an AT, a node models an atomic attack that exploits some vulnerability in the system to compromise one of the three primary security attributes, viz. Confidentiality (C), Integrity (I) or Availability (A). A directed edge models the success of the attack defined by the child node that allows the attacker to transition to a more privileged state, that is, its progress to a state that is closer to the final objective of the attacker. The presence of an arc connecting two or more edge models conjunctions, that is, *AND*ing of the success of the attacks at all children nodes of these edges. Figure 2

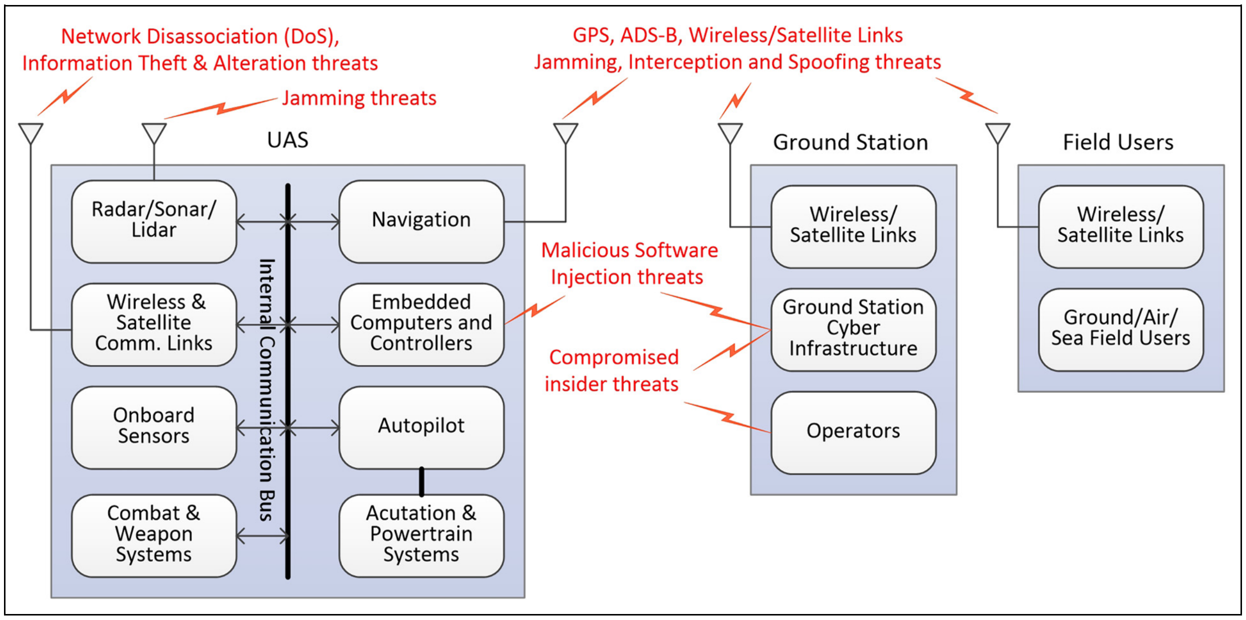
Network Disassociation (DoS),
Information Theft & Alteration threats
Jamming threats
GPS, ADS-B, Wireless/Satellite Links
Jamming, Interception and Spoofing threats
UAS
Ground Station
Field Users
Radar/Sonar/ Lidar
Navigation
Wireless & Satellite Comm. Links
Embedded Computers and Controllers
Onboard Sensors
Autopilot
Combat & Weapon Systems
Acutation & Powertrain Systems
Internal Communication Bus
Malicious Software Injection threats
Compromised insider threats
Wireless/ Satellite Links
Ground Station Cyber Infrastructure
Operators
Wireless/ Satellite Links
Ground/Air/ Sea Field Users

Figure 1. Targets (unmanned autonomous system, ground station and field users) and types of cyber threats. (Color online only.)

Table 1. Security attributes to be protected in different states of data.

Security attributes Data State	Confidentiality	Integrity	Availability
In storage	Files system, files and main memory	Files system, files and main memory	Files system, files and main memory
In processing	Leakage through side channels	Processor code	Processor code
In transit	Down link sensor data uplink control commands (wireless comm. links)	Downlink sensor data; uplink control commands; communication software code	Wireless communication links; processor and software code

shows the AT model of a simple attack situation in which the attacker's final objective is to steal confidential surveillance data gathered by an UAV (e.g. RQ-170), as denoted by the root node of this AT. Using Figure 1 as reference, the attacker will be able to achieve its objective by identifying and exploiting vulnerabilities in any one of three entities, namely, (1) the UAV, (2) the GCS or (3) the Field users, which are involved in UAV surveillance operations. The targeted data may be available in the internal memory of the UAV, at the ground station and/or it may be being communicated over tactical wireless communicated between the UAV and Field Personal Digital Devices (FPDDs).

From this model, it follows that an attacker can reach its objective, which is denoted by the root node of the AT, via any one of these paths, that is, *OR*ing three directed edges to the root node from its three ''child'' nodes, as shown in Figure 2. These ''child'' nodes are the roots of their respective subtrees, which can be reached from by *OR*ing or *AND*ing paths from their ''child'' nodes, and so on. As an example, consider the middle path. To reach the node ''Hijack UAV'', the attacker has to first reach the objective ''Gain C2 privileges'' to allow it to send unauthorized and malicious control commands to the UAV. To reach this stage, the attacker has to succeed in ''Jamming Satellite C2 link'' AND ''Spoof the GPS transmissions''. The purpose of the ''Spoof the GPS transmissions'' atomic attack will be to ensure that the attacker keeps feeding wrong GPS data to the UAV so as to prevent the UAV or its ground controller from realizing that the UAV is on a wrong trajectory.

Designing the AT model of each possible attack is the first step towards protecting a system from cyber threats. To secure a system, security mechanisms need to be designed and implemented in order to prevent an attacker from reaching the root node from its ''child'' nodes. This

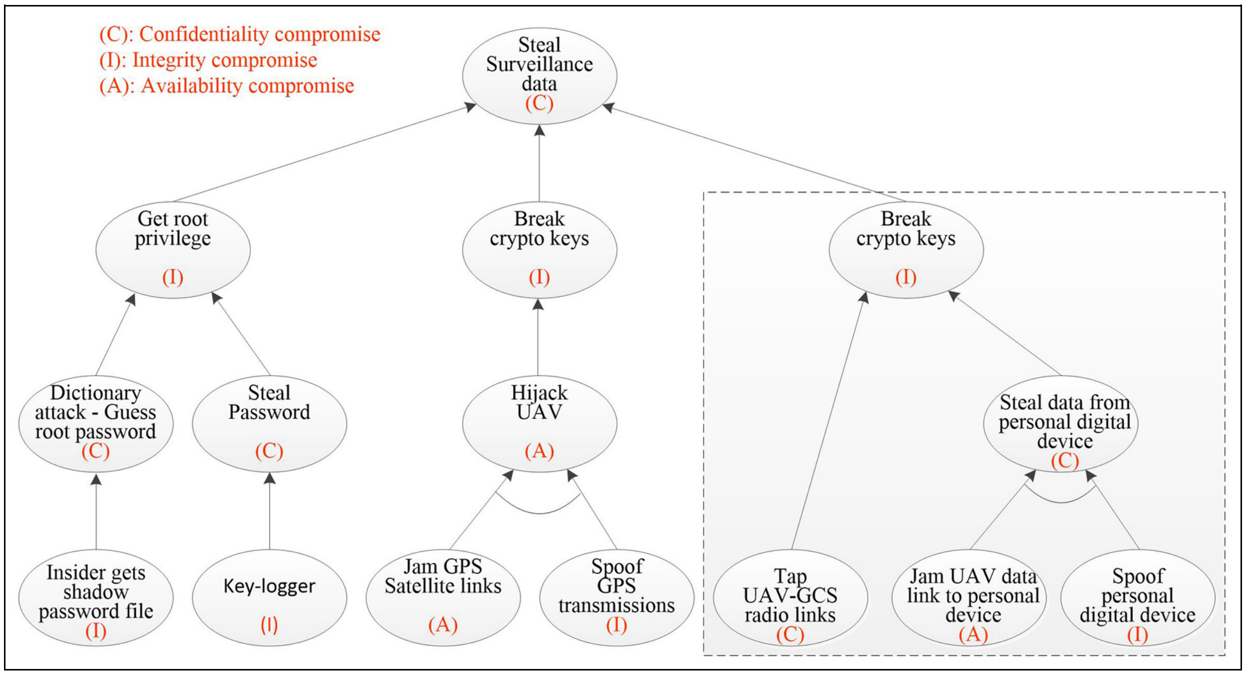
(C): Confidentiality compromise
(I): Integrity compromise
(A): Availability compromise
Steal Surveillance data (C)
Get root privilege (I)
Break crypto keys (I)
Break crypto keys (I)
Dictionary attack - Guess root password (C)
Steal Password (C)
Hijack UAV (A)
Steal data from personal digital device (C)
Insider gets shadow password file (I)
Key-logger (I)
Jam GPS Satellite links (A)
Spoof GPS transmissions (I)
Tap UAV-GCS radio links (C)
Jam UAV data link to personal device (A)
Spoof personal digital device (I)

Figure 2. Theft of unmanned aerial vehicle (UAV) confidential data—Attack Tree model.

argument is applied recursively, since such a ''child'' node is the sub-root of the AT rooted at this child node. It therefore follows that if the security mechanisms can successfully block traversal from a ''child'' node to its ''parent'' node, the attack can be stopped from reaching the root node via the path that includes this ''child'' node. Consequently, the complex problem of securing a system can be reduced to finding a solution to *n* simpler problems, where each simpler problem is one of blocking the path to the root (or the sub-root) node from its *n*''child'' nodes. As a corollary, it also follows that when the path from ''child'' nodes to their ''parent'' node involves *AND*ing, then blocking any just one *AND*ing edge is sufficient to prevent the attacker from reaching its intermediate goal of reaching the ''parent'' node.

In developing cyber security solutions, we want to emphasize that protecting a system requires security professionals to identify and enumerate all possible threats and develop AT models of all identified threats. It is also relevant to point out that it may not be mandatory to defend against all possible threats. A security solution incurs costs and it makes sense to invest in a security solution if and only if the cost of building a security mechanism is less than the cost of the asset to be protected. The issue of cost–benefit analysis is discussed in greater detail in Section 3.

Human engineered systems can and do fail due to accidents, misuse, wear-and-tear, etc. However, such failures are primarily non-malicious. Cyber systems, on the other hand, have to additionally deal with malicious failures caused by deliberate cyber-attacks. Typically, a system may contain multiple security vulnerabilities that can be converted into security threats by the attackers. However, the relevance of such threats is system and application dependent. Fixing vulnerability can incur substantial costs in developing and installing additional code in a large number of affected systems. It is therefore important to carry out a cost–benefit analysis of each potential threat and invest in defending against a threat only if the cost of the security mechanism is less than the value of the asset to be protected, as discussed in Sections 3 and 4.

3. Cyber threats to unmanned autonomous systems

UASs, particularly those used by the militaries, are becoming highly network-centric (see, for example, Figure 1). Such networked systems are designed to operate within a secure perimeter and access to the resources within this perimeter by outsider entities is tightly controlled by an outer firewall to prevent unauthorized access to these resources. In addition, since UASs perform sensitive operations and deal with sensitive data, there is also a need to ensure that only authorized insider entities should have easy access to these resources. However, we need to differentiate between two classes of insiders. The first class consists of those entities, who despite being members of the organization, are not associated with the task or the mission being performed by the UASs. Based on the basic military principle of ''least privilege'', such insiders need to be prevented access to UAS resources by creating an inner firewall,[13] as discussed in Section 5. Second classes of insider entities are those that have direct access to inside UAS resources. Since performing any mission-related operation using UASs makes it necessary to access and use data, the objective of preventing such an insider from performing inappropriate actions is achieved by defining and enforcing appropriate access control policies. The objective of such access control policies is primarily to ensure the confidentiality and integrity of data. As for protecting the availability of data, as discussed in Section 2, an attacker can compromise availability in two different ways: (1) directly through Denial of Service (DoS) or Distributed DoS (DDoS) attacks; and (2) indirectly by building an AT through one or more confidentiality and integrity atomic attacks. Sections 3.1 and 3.2 discuss details of the policies required to defend to against confidentiality and integrity threats, respectively, while Section 3.3 discusses threats that can compromise system availability and proposes possible techniques for defending against such threats.

3.1 Confidentiality threats

Threats to the confidentiality of data arise from multiple sources, internal and external. Enforcing access control policy based on the Bell–LaPadula (BLP) privacy model[16] can effectively deal with the confidentiality threats posed by internal sources. The BLP model is defined in terms of sets of labels, for example, L = {Top Secret, Secret, Confidential, Un-classified} assigned to subjects (e.g. users and operating system processes) and objects (e.g. files, memory blocks, etc.).

- Simple policy: user ''U'' is allowed to perform a ''READ'' operation on object ''O'' only if $L_U \geq L_O$. This policy ensures that a lower level user cannot divulge higher level information.
- *-Policy: user ''U'' is allowed to perform a ''WRITE'' operation on object ''O'' only if $L_U \leq L_O$. This policy ensures that a user with access to high-level information is not allowed to leak high-level information by writing to a lower level object (referred to as ''side channels'').

The BLP policy is referred to as the ''Mandatory Policy'' and its strict implementation does not allow a user to modify its security label or of any of the objects. This ensures that the system never arrives in a state in which higher level information becomes accessible to lower level users explicitly or implicitly through side channels. In contrast, many commercial operating systems (e.g. Windows, Linux, Unix, Android, MAC OS X, etc.) support the so-called ''Discretionary Policy'', which allows users to change access control to objects at their discretion, which can compromise confidentiality. Since cyber systems in UASs, their GCSs and FPDDS often use such operating systems, it becomes possible either for an inside or an outside attacker to leak information by creating side channels. (The discussion on AT in Section 2 showed that the confidentiality of an UAS can also be compromised indirectly through a sequence of other attacks.) For example, an insider with ''Secret'' level clearance can read a file containing ''Secret'' information, copy its contents into another file and then set the classification of the second file to ''Unclassified'' and make it readable for all, irrespective of their clearance level. An outside attacker can use this process to create a side channel, but it first has to intrude into the system using root kits, buffer overflow (BOVF), social engineering attacks (e.g. phishing, Cross Site Scripting, etc.) and insert malicious code to create a side channel as described earlier. As an example, a recently reported vulnerability[17] that causes stack overflow in the Android operating system used to run smart phones and other FPDDs allows the attacker to insert malicious code[18] to create a side channel and use it to steal sensitive data.

An attacker can also compromise the confidentiality of an UAS by capturing data communicated over network links. It is generally hard to capture data that is communicated over wired links, for example, an UUV being remotely controlled from its mother ship via an umbilical cable. However, in the case of UAVs and UGVs, since data invariably travels over wireless links it is relatively easy for an attacker to capture such data using inexpensive data sniffers. This data consists of control commands (in the uplink direction) and sensor or surveillance data (in the downlink direction). Since both are considered sensitive, they are often encrypted to ensure their confidentiality. Consequently, threats to UAS data confidentiality result from the weakness of data encryption techniques. For example, older WiFi links based on the Wired Equivalent Privacy (WEP) or Wireless Protected Access (WPA) technology with RC4[19] encryption use 64-bit keys and even a moderately powerful laptop computer can break RC4 quickly.[20] In contrast, the WPA2 technology employs the much stronger Advanced Encryption Standard (AES) with 128- or 256-bit keys, which is considered to be practically unbreakable and has become a standard for encrypting application data communicated over WiFi links. However, WiFi control commands used to set up (also referred to as *Association*) or tear down (also referred to as *Disassociation*) a WiFi connection between a mobile device and a WiFi Access Point (AP) or router are not encrypted. This fact, and easy availability of software tools, allows attackers to spoof the hardware or the media access control (MAC) address of the UAS's WiFi adapter and to inject WiFi disassociation control frames that appear to originate from the spoofed UAS. When the AP receives such a frame, it disconnects the spoofed UAS, forcing the UAS to re-establish connection with the AP. The connection setup or the association process requires the UAS to provide to the AP its WPA2 authentication information, that is, password. The attacker typically launches such a disassociation attack repeatedly, which in turn causes the AP to immediately initiate the four-way 802.11 WPA2 handshake for re-establishing the association with the just disassociated UAS. The association process is primarily concerned with creating a new Pre-Shared Key (PSK),[21] which is computed independently by the two communicating ends (i.e. an AP and an UAS) using the Password Based Key Derivation Function-2 (PKBDF2)[22] based on the WPA2 password phrase. The PSK in turn is used to compute other cryptography keys, including the ones used for AES encryption. Even though the PSK never travels over the wireless link as part of the four-way handshake, the Message Integrity Code (MIC) of the created PSK is exchanged between the two communicating ends and this MIC can be intercepted and used by the attacker to guess the PSK[23] and AES encryption keys derived from this PSK.

3.2 Integrity threats

Protecting the integrity of the data is concerned with the authenticity of the received data. There are two main causes of integrity compromises:

(i) data sent by the sender is modified or corrupted by the attacker before it arrives at the receiver (also referred to as "Man-in-the-Middle" (MITM) attack);
(ii) attacker assumes the identity of the sender and succeeds in sending fake data that appear to the receiver as being sent by the real sender (also referred to as "Origin Authentication").

Integrity compromises can have serious consequences, since corrupted data used by a decision-making process will result in incorrect decisions. If an attacker launches a MITM attack on the C2 data sent to an UAS, the attacker can effectively hijack its functionality. Similarly, if an attacker modifies the downlink surveillance data being used in a decision-making application supporting a military mission, the consequences can be disastrous. Like threats to data confidentiality, threats to data integrity arise from both internal and external sources. There are two main techniques for dealing with these threats—policy-based and cryptography-based techniques, detailed in the next section.

3.2.1 Integrity protection policies. The first set of integration protection policies were proposed by Biba[24] through the introduction of the notion of integrity labels. Biba's model assumes the system to consist of three entities—a set of subjects S (e.g. users and software processes), a set of objects O (e.g. files, memory blocks, etc.) and a set integrity levels I, similar to security levels L used in the BLP model. Each $s \in S$ and $o \in O$ are assigned integrity labels $i(s) \in I$ and $i(o) \in I$, respectively. Biba[24] defined the following integrity polices.

1. Strict Integrity Policy is a mandatory access control policy that satisfies the follows:
   - *Simple Integrity Property*, which requires that a subject s can read an object o if and only if $i(s) \leq i(o)$; and
   - *-Property, which requires that subject s can write to object o only if $i(s) \geq i(o)$.
2. Low Watermark Policy relaxes the read policy to allow it to read any object as follows.
   - If a subject s reads an object o with lower integrity levels, then its new integrity level $i_{\text{new}}(s)$ is reduced to that of the object, that is, $i_{\text{new}}(s) = \min(i(s), i(o))$. As a result, a subject's integrity level can keep getting reduced monotonically, eventually reaching the lowest level. This downside is remedied by periodically restoring $i_{\text{new}}(s)$ to a higher level.
   - Subject s is allowed to write to object o only if $i(s) \geq i(o)$.
3. Ring Policy further relaxes the read policy by rationalizing that the (human) user's integrity cannot be compromised by reading; only the write operation can compromise the integrity of an object as follows:
   - subject s is free to read any object o irrespective of integrity levels;
   - write policy restricts the ability of subject s to write to object o only if $i(s) \geq i(o)$.

Policies used to protect confidentiality and integrity are defined by the management of an organization that includes security professionals of an establishment and are implemented and enforced by the system administrators, who are assumed to be trustworthy. While this is generally true, there have been instances of system administrators being compromised due to political convictions, financial gains, employee disenchantment or blackmail. Most recent incidents of the compromise of the personal information of a very large number of past and present federal employees have created the risk of some of these employees being subjected to blackmail. In the long run, GCS and the FPDD segments of an UAS are likely to benefit mostly from regular audits of insiders (including system administrators). This and the policy of employing strong cryptography techniques are becoming a necessity to protect the data's confidentiality and integrity. Policy-based integrity protection works best for insider subjects, that is, authorized users or software processes whose natural behavior is not to intentionally attempt to defeat prescribed policies. In contrast, the organic behavior of an attacker, whether an insider or an outsider, is to deliberately compromise policies. The cryptographic integrity protection techniques discussed in the next section offer solutions for preventing intentional and unintentional efforts to defeat policy-based techniques.

3.2.2 Integrity protection with cryptography techniques. Cryptography techniques protect the integrity of data in all its states, viz. data at rest in memory, files or databases, in processing and while in transit over network links. MIC and *Digital Signature* (DS) are two closely related cryptographic constructs that are used for protecting data integrity. Both MIC and DS compute a fixed-sized finger print by applying a hash function or pseudo random function (PRF) $h(m)$ to the arbitrary size data block or message m to be protected. A stored data block, file and communicated message protected by MIC or DS is now a tuple $< m,\ h(m) >$. A user or a receiver of this tuple computes its own hash value, say, $h'(m)$, and m is considered to have preserved its integrity if and only $h'(m) = h(m)$. Acceptable hash functions need to satisfy the following properties:

- $h(m)$ output of a fixed number of bits (typically $|h(m)| < < |m|$, where $|\cdot|$ denotes the number of bits;
- first image preresistance;
- second image preresistance;
- collision resistance.

The last three properties ensure that for a given $h(m)$, it is computationally infeasible for an attacker to fabricate some other message m' such that $h'(m) = h(m)$. MD5[25] was the most commonly used MIC. However, although it has been formally shown to be breakable[26,27] it is still in use. It should also be realized that MICs, such as the MD5, do not provide any data or message origin integrity. Lack of origin integrity presents a serious problem for ensuring the integrity of data that is either shared by multiple entities or travels across network links. Consider an example situation in which the enemy has the ability to jam an UAV, spoof its network identity and listen to the surveillance/reconnaissance data m being sent by an UAV to a FPDD of friendly forces using this data for their mission. The enemy can now substitute m with some fake data m_{fake} and create its MIC $h(m_{\text{fake}})$ to make it look like it was sent by the authoritative UAV, thus putting the entire mission and lives of friendly forces at dire risk.

These limitations can be overcome by two techniques. For example, the Internet protocol secure-authentication header (IPSEC-AH)[28] includes the hash of the entire Internet protocol (IP) packet (IP header + message m). Since the IP header contains the sender's IP address, the receiving end is able to verify the authenticity of both the sender and the message m. The second set of techniques use a secret key k for computing the hash value $h(m,\ k)$ by making it a function of both m and k. The secret key k can be incorporated into $h(m,\ k)$ in a variety ways.[29] The Secure Hash Algorithm (SHA)[30] and Hash-based Message Authentication Code (HMAC)[31] are examples of this technique. Adding k to the hash implies that the sender of $< m,\ h(m,\ k) >$ has to securely share k with the message receiver. Moreover, the receiver has to trust that k is indeed owned by the entity that claims to be sender, to enable the receiver to verify the integrity of m and its origin.

The problem of trust between the originator and the receiver just alluded to can be overcome through the use of the Digital Signature Algorithm (DSA)[32,33] and Elliptic Curves DSA[33,34] that use public key cryptography (PKC) techniques.[35–38] Instead of having to share the secret key k, the PKC uses a pair of keys $\langle d, e\rangle$, where d is a secret known only to its owner, and e is made public. The association between d and its owner is established using Digital Certificates (DCs), such as the X.509 standard and the Public Key Infrastructure technology.[39] Therefore, to enhance the dependability of UASs, highly sensitive information, for example, critical commands sent to the UAS, such as self-destruct, fire a missile, etc., should be digitally signed to minimize the risks posed by MITM threats.

Defining and implementing appropriate policies, employing cryptographic techniques, or both, are necessary but not sufficient to prevent integrity attacks. As discussed in Section 2, an attack typically involves a sequence of several atomic attacks that can be linked together as an AT. Consequently, defending the integrity of UASs will require cyber security personnel to design all possible ATs whose root nodes model some integrity compromise. As an example, consider the subtree marked by the dotted rectangle in Figure 2. If an attacker reaches the root node ''Break crypto keys'' of this subtree and succeeds in accessing the shared secret k, it can then take the step of creating signed but malicious control commands or altering surveillance/reconnaissance data.

3.3 Availability threats

Protecting the *Availability* of a system or data is concerned with ensuring that attackers do not succeed in preventing authorized entities from on-demand and timely access to data and system services. UASs utilize data (i.e. tele or autonomously computed commands) and cyber services for real-time control of motion, onboard engines, various control surfaces and sensors that provide surveillance/reconnaissance data that is often used to support missions having real-time constraints. Conversely, an attacker's objectives are to crash software components that are used to generate autonomous control commands, prevent timely delivery of remote commands sent over wireless or satellite links to an UAS and/or prevent timely delivery of surveillance/reconnaissance data from an UAS. Therefore, threats that can compromise the Availability of UAS include the following.

- Jamming data communication links: the UAV and UGV use wireless radio links to communicate with the GCS and to communicate surveillance and reconnaissance data. Similarly, UUVs may use acoustic communication links for their operations. Communication links, whether radio or acoustic, can be jammed by the jammers. A jammer transmits a high-energy random noise signal to increase interference (or greatly reduce the signal-to-inference ratio (SIR)) with the objective of increasing the bit error rate to unacceptable levels.[40] Spread-Spectrum digital communication techniques (e.g. Direct Sequence Spread Spectrum and Frequency Hopping Spread Spectrum (FHSS)) are commonly used to minimize the adverse impacts of jamming.[40]
- DDoS attacks on GCS networks and FPDDs: communication between the GCS and UASs often employ long-haul networks consisting of the interworking of multiple network segments, including publically accessible Internet segments. Attacking the Domain Name Service (DNS) by poisoning locally cached DNS records[41] has been used successfully to disrupt the normal functioning of the Internet at a global level.[42] Flooding a server or a network link with useless traffic is another common technique employed by attackers to compromise Availability. Typically, more sophisticated flooding attacks are DDoS attacks using botnets.[43] A botnet consists of a large number of hijacked ordinary home computers, which are commanded by the attacker to simultaneously generate meaningless traffic directed at a targeted GCS server or Internet-visible FPDDs.
- Injecting malware (viruses, worms and Trojans) into GCS computers or FPDDs: since the GCS, UASs and FPDDs are all interconnected, malware injected into the GCS or a FPDD can also travel to UASs. Until 2010, BOVFs in the stack and the heap areas of the main memory were one of the commonest vulnerabilities that were exploited by the attacker for injecting malware. The stack BOVF exploits can be easily written to cause a return address being overwritten with random bytes to destroy a valid return address. This will typically cause a running program to try to access memory not allocated to it and force the software application containing such code to be immediately terminated by the operating system. Tools, for example, StackGuard,[44] StackOFFence,[45] etc. were developed to mitigate such stack smashing BOVF exploits. Finally, the use of a ''No Execute'' (NX) bit applied to the stack segment in the Windows-XP SP2 and to the Intel X86 class of processors ensured that any bytes placed on a stack cannot be interpreted as executable code. This fix led to a sharp reduction in such exploits. However, NX restrictions placed on the stack can be overcome by the use of the Return Oriented Programming (ROP) concept[46,47] and opening another venue of BOVF exploits.[48] Similarly, BOVFs in buffers residing in the heap areas of memory have created serious vulnerabilities in systems ranging from handheld mobile devices[49,50] to high-end servers[51,52] used in servers. Exploitation of these vulnerabilities can cause some GCS servers, UAS embedded computers and handheld FPDDs to malfunction and compromise system Availability.

4. Threat modeling and risk analysis

Threat modeling is the first step towards building secure systems.[53] It is used to identify vulnerabilities and find how attackers can potentially exploit these vulnerabilities in the system to be secured. In Section 2, we informally highlighted the need for the divide-and-conquer approach for managing security of complex systems by identifying various systems and subsystems (components) and interactions (inputs and outputs) amongst these components and with the external environment. In fact, Chapter 10 in Howard and LeBlanc[53] is titled ''All Input Is Evil'' to emphasize the point that every input received by a component from the outside can pose a threat to the system. We take this assertion a step further by emphasizing that in the case of UASs, any output sent to the external environment also poses a threat. Outputs can not only be observed by unauthorized entities, but they can also be subjected to malicious alterations before reaching their destination.

According to Howard and LeBlanc,[53] threats to information systems can categorized into six generic types – Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service and Elevation of privilege (STRIDE). The first four threats (Spoofing identity, Tampering with data, Repudiation, Elevation of privilege) are synonymous with *Integrity*. Information

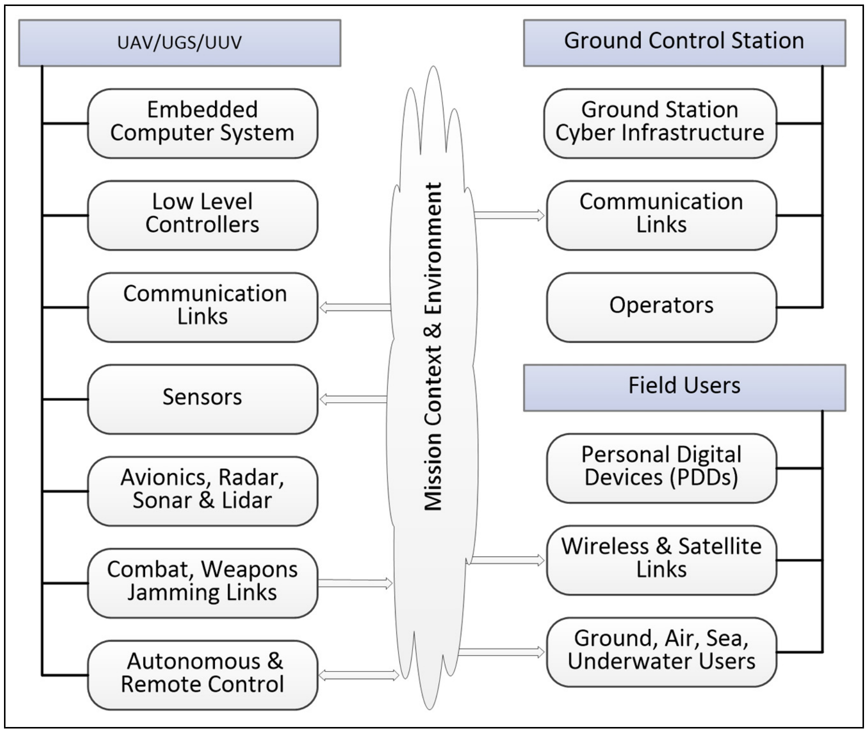
UAV/UGS/UUV
Embedded Computer System
Low Level Controllers
Communication Links
Sensors
Avionics, Radar, Sonar & Lidar
Combat, Weapons Jamming Links
Autonomous & Remote Control
Mission Context & Environment
Ground Control Station
Ground Station Cyber Infrastructure
Communication Links
Operators
Field Users
Personal Digital Devices (PDDs)
Wireless & Satellite Links
Ground, Air, Sea, Underwater Users

Figure 3. Data flow between unmanned autonomous system (UAS) components.

disclosure threats compromise the *Confidentiality* and DoS threats compromise the *Availability* of system functionalities. Recent research papers[54,55] demonstrate the applicability of the STRIDE approach to model threats and the use of these models for evaluating risk posed by these threats to CPSs and UAVs. The STRIDE approach suggests first dividing a system into functional components to facilitate identification of various threats that have the potential to compromise functionality of these components. Figure 1 shows the application of this step to the UAS infrastructure used in typical military applications. Since data is the primary resource that is used and can be compromised by cyber threats, the second step of threat modeling is to develop the so-called Data Flow Diagram (DFD), which identifies all possible data flows (i.e. inputs and outputs) between these components and from/to the external environment. Figure 3 shows the DFD for the UAS infrastructure shown in Figure 1.

After identifying data flows, the next step is to analyze the cyber risks posed by threats that can adversely affect Confidentiality, Integrity and Availability of these data flows. There are several risk analysis tools are available, which include the CVSS,[56] OCTAVE,[57] MEHARI[58] and ETSI.[59] In this paper, we use the CVSS technique to evaluate the cyber risks to the UAS exemplified in Figure 1. It associates a risk with vulnerabilities based on the following three metrics groups – Base Group, Temporal Group and Environmental Group.

The Base Group assigns a risk score to a vulnerability based on its exploitability, which depends on the following:

1. Access Vector (AV)—(Local, score = 0.395); (Adjacent Network, 0.646); (Public Network, 1.0);
2. Access Complexity (AC)—(High, score = 0.35); (Medium, 0.61); (Low, 0.71);
3. Authentication (Au)—(Multiple Times, score = 0.45); (Single, 0.55); (None, 0.705).

In addition, since a vulnerability sooner or later materializes into a threat, the CVSS also assigns an Impact metric to a vulnerability based on this threat's ability to cause *Confidentiality* (C), *Integrity* (I) and *Availability* (A) damage. The Impact metric and associated score is defined for each of these security attributes as (None, Score = 0), (Partial = 0.0.275), (Complete = 0.66).

The AV, AC and Au scores are combined to empirically calculate *Exploitability* and *Impact* sub-scores as

$$Exploitability = 20 \times \text{AV} \times \text{AC} \times \text{Au} \tag{1}$$

$$\begin{aligned} Impact &= 10.41 \\ &\times [1 - (1 - Impact_C)(1 - Impact_I)(1 - Impact_A)] \end{aligned} \tag{2}$$

$$f(Impact) = 0 \text{ if} Impact = 0, \text{ else } f(Impact) = 1.17 \tag{3}$$

Table 2. Risk score to an unmanned autonomous system from different assumed threats.

Threat	AV	AC	Au	*Impact$_C$*	*Impact$_I$*	*Impact$_A$*	B-Secore
C2 radio link jamming	Adj Net	Low	None	None	None	Comp.	5.4020
Data link snooping	LocNet	High	Mult	Comp.	None	None	3.6691
MITM Command Link	PubNet	High	Mult	None	Part.	Part.	3.2021
MITM Data Link (local)	LocNet	High	Mult	None	Comp.	None	3.6691
MITM Data Link (Pub)	PubNet	High	Mult	None	Comp.	None	4.5656
Malware infection	PubNet	High	Mult	Part.	Part.	Part.	4.2639
KeyLogger	LocNet	High	Mult	Comp.	Part.	Part.	4.8538
WiFi DisAssociate (DoS)	LocNet	Low	None	None	None	Comp.	4.9413
WiFi DisAssociate (KeyTheft)	LocNet	High	Single	Comp.	Part.	Part	5.1842

AV: Access Vector; AC: Access Complexity; Au: Authentication; C2: command and control; MITM: Man-in-the-Middle; DoS: Denial of Service.

The sub-scores are combined to arrive at the *Base Score* for vulnerability using the empirical formula:

$$Base\,Score = f(Impact) \times [(0.6 \times Impact) + (0.3 \times Exploitability) - 1.5)] \quad (4)$$

Together with the score, the CVSS technique assigns a base score vector to each vulnerability and threat:

$$Base\,Score\,Vector = [BaseScore, \mathrm{AV}, \mathrm{AC}, \mathrm{Au}, Impact_C, Impact_I, Impact_A] \quad (5)$$

The CVSS approach for calculating the risks faced by an UAS from a set of exemplary threats is summarized in Table 2.

In this example situation, we have assigned rational but ''guesstimated'' values to different threat parameters, that is, AV, AC, Au and impact of a threat to system confidentiality, integrity and availability, that is, $Impact_C$, $Impact_I$ and $Impact_A$, respectively. For example, the ''C2 Radio Link Jamming'' threat is assumed to be launched from a jammer that is part of the enemy network whose coverage extends to the targeted UAS. Assuming that the UAS C2 link employs a sophisticated FHSS anti-jamming measure, the ''Access Complexity'' (AC) is set ''Low''; the jammer only needs to transmit a high-powered noise signal that occupies the frequency band of the C2 radio link. A further ''Au'' entry is set to ''None'', since jamming does not require any authentication. The last two rows of Table 2 calculate the risk posed by the WiFi DisAssociate vulnerability that is inherent to any WiFi link (including WiFi links that use WPA encryption), since control commands in a WPA WiFi are not sent with any encryption. The WiFiDisAssociate attack only requires an attacker to send a DisAssoc data frame containing the spoofed Ethernet address of a WiFi node and requires relatively low complexity. However, its impact is to only complete compromise *Availability*. In contrast, the WiFiDisAssociate (KeyTheft) attack[60] is complex, requiring sophisticated crypto analysis. However, it is capable of complete compromise of link *Confidentiality*.

The B-Score (*BaseScore*) column quantifies security risks that an UAS faces from different threats and is typically used to prioritize threats. Since managing any threat requires resources (e.g. CPU cycles, memory, communication bandwidth, financial budget, etc.), which are invariably limited, threat prioritization based on formal risk analysis facilitates allocation of resources objectively in managing identified threats. For the situation exemplified in Table 2, employing a jam-resistant C2 radio link (typically using FHSS technology) assumed the highest priority. Table 2 also identifies high risk that is inherent in existing WiFi technology because of the unencrypted control channel. Therefore, risk analysis also points out that finding a suitable fix for this vulnerability is a precondition for WiFi to be employed in an UAS for communicating surveillance and reconnaissance data securely.

The primary effect of this threat is to cause a DoS attack, that is, complete *Availability* compromise. The attacker can also eavesdrop on the four-way WPA handshake data exchanges and, with some additional effort, break the WPA passphrase using dictionary attacks, thus causing the *Confidentiality* attribute to be partially compromised. (By *partial*, we imply that attacker may not succeed in stealing WPA passphrases of all WiFi nodes.) With some more effort, spoofed WiFi data frames can also be inserted (i.e. a MITM attack) to compromise *Integrity* of this WiFi Link. Plugging in the CVSS scores mentioned above in Equations (1)–(5) gives $Exploitability = 10$, $Impact = 8.4$ and $Base\,Score = 8.9$.

In addition to computing to the *BaseScoreVector*, the CVSS can also capture the effects of time (e.g. with time, more software bugs and vulnerabilities get fixed) and environmental factors (e.g. collateral damage, location/ nationality sensitive passwords) on the score vector by defining Temporal Group and Environment Group metrics, as shown in Figure 4.[61]

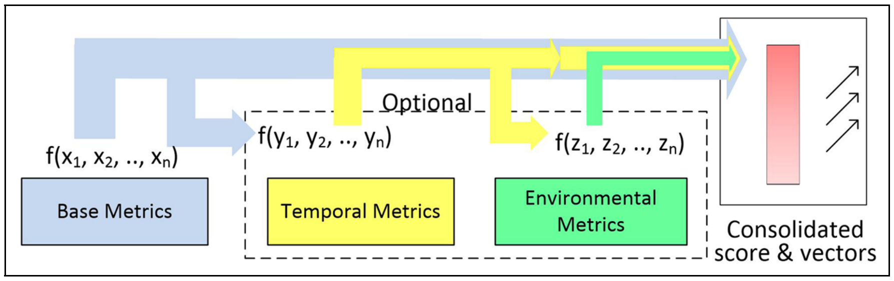

Figure 4. Common Vulnerability Scoring System scoring groups.

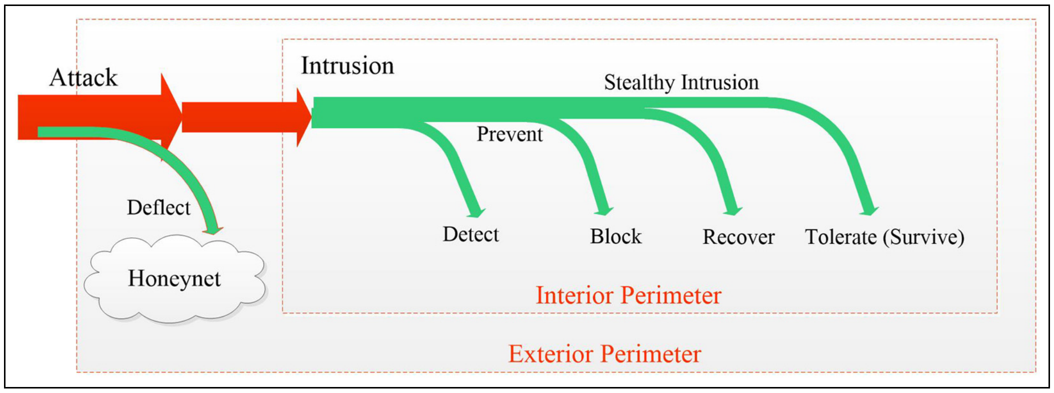

Figure 5. Attack handling stages.

5. Defending against cyber threats to unmanned autonomous systems

Based on our interpretation of the presentation on the Air Force's medium term Cyber Vision,[62] protecting UASs from cyber threats, particularly those used in military applications, requires multi-stage defensive architecture, as shown in Figure 5. In this diagram, the "Attack" is used to describe malicious attempts to intrude into the inner perimeter. The term "Intrusion" is used to represent an event that indicates that the outer perimeter has failed in detecting and preventing the attack, which now manifests as an intrusion into the inner perimeter. The outer perimeter's purpose is to predict and detect attacks perceived to be directed towards the Air Force systems, including its UAV assets, and try to deflect such attacks. Attack deflection is achieved by placing a honeynet[63] in the outer perimeter. The honeynet uses cyber maneuvers[64] to divert attacks to the nodes within the honeynet. However, attack deflection mechanisms cannot be guaranteed to always succeed with probability one. When attack deflection fails, such an attack becomes an intrusion into the inner perimeter. Simultaneously, systems within the inside perimeter need to be subjected to proactive identification of vulnerabilities using malware scanning, probing and penetration testing using tools (e.g. NMAP,[65] METASPLOIT,[66] SNORT,[67], system audits, etc.). Vulnerabilities so identified need to be removed by applying hardware, software and policy fixes to minimize intrusion probability. It should also be emphasized that scanning, probing and penetration testing efforts can fail to expose certain vulnerabilities; some identified vulnerabilities remain unpatched or attackers race ahead to uncover vulnerabilities. Such vulnerabilities are termed as *zero day vulnerabilities*[68] and have become a serious cyber security concern, as evidenced by the *Heartbleed*[69] security bug attack on the widely used *openSSL* used to secure Internet communication, which may also be used by GCSs, UASs and PDDs.

When an attack becomes an intrusion its mitigation starts with intrusion detection through the use of Intrusion Detection Systems (IDSs).[70] IDSs are either signature-based (e.g. SNORT[67]) or anomaly based (e.g. PAYL,[71] POSEIDON[72]). An IDS typically issues only a notification that an intrusion has occurred and it does not take any action to mitigate the intrusion. An Intrusion Prevention System (IPS), besides including the functionality of the IDS, has additional logic for taking actions against an inclusion. For example, most retail malware scanners installed on home and office computers, besides detecting a malware in a downloaded file, also block or quarantine such files. Similarly, a network firewall installed at the entry point of interior perimeters implement logic for preventing the entry of packets that are deemed to be malicious. For the UAS infrastructure consisting of the GCS, UASs and FPDDs, although expected to be well protected by IDSs and IPSs, such protection tools have a small but finite probability of failing against sophisticated intrusions. When such failures occur in home, office and eCommerce computing, the common practice is to try to recover from the damage caused by intrusions. For example, if the intrusion resulted in the theft of personal information, recovery actions will include forensic analysis to identify the vulnerabilities exploited, developing an AT model of the intrusion, taking counter measures and informing affected individuals.

Recovery actions as identified above often take an indeterminate amount of time. Clearly, therefore, such a reactive strategy of responding to an intrusion is not workable for systems such as the UASs that have real-time constraints, where failure to meet these constraints can have serious implications. The alternative is to develop cyber system architectures that are organically intrusion tolerant and can survive intrusions. The basic objective of intrusion tolerance is to allow a cyber system to continue providing its services, despite having been compromised by intrusions. The Intrusion Tolerance (IT) idea was motivated by the principles of Fault Tolerant Systems (FTSs) that are used in applications requiring high availability (e.g. aircraft flight control systems). FTSs are designed primarily to deal with non-malicious failures caused by wear and tear, accidents, extreme conditions, human failures, etc., and tolerance to faults is typically achieved by incorporating redundancy in the system design.[73] IT, in contrast, is motivated by the need to minimize the adverse impacts of security failures caused by malicious intrusions, which cannot only compromise system availability, but also system confidentiality and integrity. In the past few years, several intrusion tolerant cyber-system architectures have been proposed. A survey of these architectures is available in Heo et al.[74] and three of these architectures, namely SITAR,[75], MAFTIA[76] and SCIT,[77] have been compared by Nguyen and Sood.[78] A quantitative security model developed by Madan et al.[79] has shown that the security performance in terms of the probability of arriving into security failed states and Mean-Time-To-Security-Failure (MTTSF) to reach these states is improved in intrusion tolerant architectures. However, none of these architectures are capable of simultaneously tolerating confidentiality, integrity and availability-centered intrusions, making these architectures unsuitable for UASs. The simple redundancy used in SITAR,[75] MAFTIA[76] and SCIT[77] in fact increases the probability of confidentiality and integrity compromises.[80]

The architecture proposed by Madan and Banik[80] is based on data Fragmentation, Coding, Dispersal and Reassembly (FCDR) of coded fragments across multiple, diverse and non-overlapping systems, and has been shown to be capable of surviving all such intrusions. In this architecture, each word (of word-length w = 8, 16, 32 or 64 bits) of a data block or a packet is fragmented into n fragments, such that each fragment is v-bits long, where $v = w/n$. These n fragments are augmented by k additional fragments of the same bit-size using erasure coding (e.g. Reed–Solomon coding[81]) that creates a new word of $n + k$ fragments. These fragments are stored in $n + k$ separate storage blocks. Similarly, the data frame to be communicated over a wireless link is also transformed into $n + k$ fragments, which can be communicated using FHSS techniques.[82] Since each fragment is v-bits long and a fragment can assume any random value $x, 0 \leq x \leq 2^{v-1}$, such that value x is transmitted at a different carrier frequency f_x. Ensuring that the mapping $x \rightarrow f_x$ is secret and known only to authorized senders and receivers, an attacker listening to these transmissions has no way of knowing the current carrier frequency f_x, thus making it very hard for the attacker to jam these transmissions. Similarly, not knowing f_x, the attacker will not be able to steal or spoof the data being communicated. This FHSS technique is referred to as Slow-FHSS. Fast-FHSS (F-FHSS) is a more sophisticated version of FHSS that uses multiple carrier frequencies within each bit time interval.[82]

Erasure codes have the fundamental property that any n healthy fragments out of $n + k$ fragments are necessary to arrive to be reassembled into the original data word. It therefore follows that a FCDR-based cyber system has the ability to survive up to $n - 1$ intrusions designed to compromise confidentiality and at least k successful intrusions designed to compromise integrity and availability.[80] The ability of the FCDR-based intrusion tolerant architecture to survive multiple intrusions that can compromise the confidentiality, integrity and availability of data stored in the file systems or communicated over networks makes it suitable for building intrusion tolerance capabilities in UAS infrastructures.

6. Conclusions

UASs are being designed with insufficient attention to cyber security concerns. This is despite the fact that UASs themselves and many important applications are driven by surveillance, reconnaissance and other situational data provided by the UASs. This paper emphasizes the need for a holistic approach that can provide robust security for protecting data and cyber systems from complex threats. The proposed approach suggests starting with the identification of threats relevant to the software and hardware components that form the UAS infrastructure to be protected. We have argued that since a large number threats have to be dealt with and it is not feasible to build defenses against all such threats, cyber risk analysis should be performed to prioritize identified threats. Specifically, we have discussed the applicability of the STRIDE threat modeling and CVSS risk analysis tools for this purpose. The paper also argues for the need for modeling cyber-attacks before attempting to defend against such attacks. This need is justified by the fact that contemporary attacks can be very complex. In order to understand the behavior of a complex attack, an attack needs to be broken into a connected sequence of smaller compromises or atomic attacks, each of which exploits a single vulnerability in the system, as suggested in the well-researched "AT" approach for modeling a complex attack. The AT model can also provide insights into an attacker's behaviors, which opens up the possibility of creating attacker behavior models. The paper also presents a structured approach for defending against atomic and complex intrusions into UASs. The suggested approach calls for attacks on critical cyber systems to be dealt with in multiple stages involving attack deflection, intrusion detection, intrusion prevention and intrusion recovery, since UASs often need to be controlled in real time and have to support real-time applications. In order to meet such requirements, we highlight the need for an UAS infrastructure to proactively survive intrusions since intrusion detection, prevention and recovery can take an indeterminate amount of time. Consequently, future UASs need to be embedded with cyber systems based on architectures that are inherently intrusion tolerant. This paper suggests the use of the FCDR technique due to its ability to tolerate Confidentiality, Integrity and Availability compromises in a single architecture. We are currently developing stochastic finite state models based on Madan et al.[79] to analyze and quantify attack tolerance abilities of the FCDR techniques and other intrusion tolerance architectures[78] in order to justify their usage in UASs.

Acknowledgment

The authors acknowledge the constructive comments and suggestions made by the reviewers, which helped improve the quality of the paper.

Funding

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

References

1. Colomina I and Molina P. Unmanned aerial systems for photogrammetry and remote sensing: a review. *ISPRS J Photogramm Remote Sens* 2014; 92: 79-97.
2. Marketsandmarkets.com. *Unmanned aerial vehicle (UAV) market by class (small, tactical, strategic, special purpose), subsystem (data link, GCS, software), application (military, commercial, homeland security), funding (procurements, RDT&E, O&M), & by payload*. Dallas, TX: Marketsand Markets, 2014.
3. Frost and Sullivan, Inc. *Study analyzing the current activities in the field of UAVs*. European Commission, 2008.
4. WinneFelf JJ and Kendall F. Unmanned systems integrated roadmap. US Department of Defense, http://archive.defense.gov/pubs/DOD-USRM-2013.pdf (2013, accessed 6 March 2015).
5. DARPA. Urban grand challenge. (DARPA), http://archive.darpa.mil/grandchallenge/ (2007, accessed 20 November 2013).
6. Guizzo E. How Google's self-driving car works. IEEE Spectrum, http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works (18 October 2011, accessed 8 December 2013).
7. Barry B. Rise of the machines? The increasing role of Unmanned Ground Systems in land operations. IISS Voices, https://www.iiss.org/en/iiss%20voices/blogsections/iiss-voices-2014-b4d9/june-4703/rise-of-the-machines-bca7 (June 2014, accessed 28 June 2014).
8. Board NS. *Naval mine warfare: operational and technical challenges for naval forces*. Washington, DC: National Academy Press, 2001.
9. Navy UD. The navy unmanned surface vehicle (USV) master plan. US Department of Navy, http://www.navy.mil/navy data/technology/usvmppr.pdf (23 July 2007, accessed 31 January 2015).
10. Jaffe G and Erdbrink T. Iran says it downed U.S. stealth drone; Pentagon acknowledges aircraft downing, https://www.washingtonpost.com/world/national-security/iran-says-it-downed-us-stealth-drone-pentagon-acknowledges-aircraft-downing/2011/12/04/gIQAyxa8TO_story.html?wprss=rss_national-security (5 December 2011, accessed 31 January 2015).
11. Shachtman N. Computer virus hits U.S. drone fleet, *Wired*, http://www.wired.com/2011/10/virus-hits-drone-fleet/ (7 October 2011, accessed 31 January 2015).
12. Cyber Security Research Alliance. Designed-in cyber security for cyber-physical systems - workshop report by the cyber security research alliance. Cyber Security Research Alliance, http://www.cybersecurityresearch.org/documents/CSRA_Workshop_Report.pdf (3 April 2013, accessed 31 January 2015).
13. Bishop M. *Computer security - art and science*. New York: Addison-Wesley, 2002.
14. Gollmann D. *Computer security*. 3rd ed. Chichester, UK: John Wiley, 2011.
15. Schneier B. Attack trees – modeling security threats. *Dr Dobbs J* 1999; 24: 1–9.
16. Bell D and LaPadula L. Secure computer systems: mathematical foundations, http://www.albany.edu/acc/courses/ia/classics/belllapadula1.pdf (March 1996, accessed 12 January 2013).
17. Database NV. National vunerability database. US-CERT/NIST, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-3100 (7 February 2014, accessed 6 April 2015).

18. Hay R and Avi Dayan A. Android keystore stack buffer overflow, https://www.intelligentexploit.com/articles/Android-KeyStore-Stack-Buffer-Over%EF%AC%82ow.pdf (2014; accessed 2 December 2015).
19. Rivest R and Schuldt JC. Spritz—a spongy RC4-like stream cipher and hash function, http://people.csail.mit.edu/rivest/pubs/RS14.pdf (2014, accessed 7 April 2015).
20. AlFardan N, Bernstein DJ, Paterson KG, et al. On the security of RC4 in TLS and WP. In: *USENIX security symposium*, Washington, DC, http://www.isg.rhul.ac.uk/tls/RC4biases.pdf (2013, accessed 7 April 2015).
21. Organization IS. IEEE 802.11i, http://standards.ieee.org/getieee802/download/802.11i-2004.pdf (23 July 2004, accessed 12 April 2015).
22. Kalislki B. Password based cryptography specification version 2.0 (RFC 2898). IETF, 2000.
23. Tsitroulis A, Lampoudis D and Tsekleves E. Exposing WPA2 security protocol vulnerabilities. *Int J Inf Comput Secur* 2014; 6: 93-107.
24. Biba K. *Integrity considerations for secure computer systems*. Technical Report MTR-3153. Badford, MA: Mitre Corporation, 1977.
25. Rivest R. RFC 1321: The MD5 message-digest algorithm (RFC 1321). IETF, 1992.
26. Dobbertin I. The status of MD5 after recent attacks. *Cryptobytes* 1996; 2: 1-6.
27. Wang X and Yu H. How to break MD5 and other hash functions. In: *Proceedings of the 24th annual international conference on theory and applications of cryptographic techniques, (EUROCRYPT'05)*, 2005.
28. Doraswamy N and Harkins D. *IPSec: the new security standard for the internet, intranet and virtual private networks*. 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2003.
29. Kak A. Lecture Notes on "Computer and Network Security" (Lecture 15: Hashing for Message Authentication), https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture15.pdf (2014, accessed 24 September 2013).
30. NIST. Secure Hash Standards (SHS): NIST, Federal Information Processing Standards Publication FIPS 180-3, http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf (August 2015, accessed 29 August 2015).
31. Bellare M, Canetti R and Krawczyk H. Keying hash functions for message authentication. In: *Proceedings of the 16th annual conference on advances in cryptology*, London, 1996.
32. El Gamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans Inform Theor* 1985; 31: 468-472.
33. NIST. Digital Signature Standard (DSS) IFIP 186-4, http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf (July 2013, accessed 29 August 2015).
34. Johnson D, Menezes A and Vanstone S. The elliptic curve digital signature algorithm. *Int J Inf Secur* 2001; 1(1): 36–63.
35. Merkle R. *Secrecy, authentication, and public key systems*. Doctoral Dissertation, Stanford University, 1979.
36. Rivest R, Shamir A and Adleman L. A method for obtaining digital signtaures and public-key cryptosystems. *Commun ACM* 1978; 21: 120-126.
37. Diffie W and Hellman M. New directions in cryptography. *IEEE Trans Inform Theor* 1979; 22: 644-654.
38. Blake I, Seroussi G and Smart N. *Elliptic curves in cryptography*. Cambridge, UK: Cambridge University Press, 1999.
39. Adams C and Lloyd S. *Understanding PKI: concepts, standards, and deployment considerations*. New York: Addison-Wesley, 2002.
40. Poisel R. *Modern communications jamming principples and techniques*. London: Artech House, 2004.
41. Stewart J. DNS cache poisoning – the next generation, https://www.ida.liu.se/~TDDC03/literature/dnscache.pdf (2003, accessed 30 August 2014).
42. Kaminsky D. Black Ops 2008: it's the end of the cache as we know it, https://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Kaminsky/BlackHat-Japan-08-Kaminsky-DNS08-BlackOps.pdf (2008, accessed 30 August 2014).
43. Nazario J. Botnet tracking: tools, techniques and lessons learnt, https://www.blackhat.com/presentations/bh-dc-07/Nazario/Presentation/bh-dc-07-Nazario.pdf (2007, accessed 30 August 2014).
44. Cowan C, Pu C, Maier D, et al. StackGuard: automatic adaptive detection and prevention of buffer-overflow attacks. In: *Proceedings of the 7th USENIX security symposium*, San Antonio, Texas, 1998.
45. Madan B, Phoha S and Trivedi K. StackOverflow fence: a technique for defending against buffer overflow attacks. *J Inform Assur Secur* 2006; 1: 129-136.
46. Designer S. Bugtrack: getting around non-executable stack (and fix), http://seclists.org/bugtraq/1997/Aug/63 (10 August 1997, accessed 30 August 2014).
47. Shacham H. The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86). In: *Proceedings of the 14th ACM conference on computer and communications security*, Alexandria, VA, 2007.
48. US-CERT. US-CERT/NIST CVE-2012-4969. National Vulnerability Database, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-4969 (2012, accessed 30 August 2014).
49. US-CERT. National Vulnerability Database: US-CERT/NIST Advisory CVE-2014-3100, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-3100 (2 July 2014, accessed 2 July 2014).
50. Hay R and Dayan A. National Vulnerability Database: US-CERT/NIST CVE-2014-3100, android keystore stack buffer overflow, https://www.exploit-db.com/docs/33864.pdf (23 June 2014, accessed 23 June 2014).
51. US-CERT. National Vulnerability Database: US-CERT/NIST CVE-2014-6271, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-6271 (24 September 2014, accessed 24 September 2014).
52. US-CERT/NIST Advisory-2015-0235. National Vulnerability Database, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-0235 (2015, accessed 2 January 2015).
53. Howard M and LeBlanc D. *Writing secure code*. 2nd ed. Redmond, WA: Microsoft Press, 2003.
54. Yampolskiy M, Peter Horvath P, Koutsoukos XD, et al. Systematic analysis of cyber-attacks on CPS – Evaluating

applicability of DFD-based approach. In: *IEEE 5th international symposium on resilient control systems (ISRCS)*, Salt Lake City, UT, 2012.

55. Hartmann K and Steup C. The vulnerability of UAVs to cyber attacks - an approach to the risk assessment. In: *Proceedings of the 5th international conference on cyber conflict*, 2013.
56. FIRST. Common Vulnerability Scoring System v3.0: user guide, http://www.first.org/cvss/cvss-v30-user_guide_v1.1.pdf. (Accessed 2 February 2014).
57. Caralli R, Stevens J, Young L, et al. Introducing OCTAVE Allegro: improving the information security risk assessment process, http://resources.sei.cmu.edu/asset_files/TechnicalReport/2007_005_001_14885.pdf (May 2007, accessed 2 February 2014).
58. clusif. MEHARI 2010: fundamental concepts and functional specifications, https://www.clusif.asso.fr/fr/production/ouvrages/pdf/MEHARI-2010-Principles-Specifications.pdf (August 2010, accessed 2 September 2013).
59. ETSI Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON). ETSI TR 101 771 V1.1.1, http://www.etsi.org/deliver/etsi_tr/101700_101799/101771/01.01.01_60/tr_101771v010101p.pdf (2004, accessed 24 September 2014).
60. Crack Wi-Fi with WPA/WPA2-PSK using Aircrack-ng. http://www.shellhacks.com/en/Crack-Wi-Fi-with-WPA-WPA2-PS K-using-Aircrack-ng (2014, accessed 2 February 2014).
61. Mell P, Scarfone K and Romanosky S. A complete guide to the common vulnerability scoring system, http://www.first.org/cvss/cvss-v2-guide.pdf (2007, accessed 2 February 2014).
62. Maybury M. Air Force Cyber Vision 2025. In: *Proceedings of the 5th international symposium on resilient control systems (ISRCS)*, Salt Lake City, UT, https://secure.inl.gov/isrcs2012/Presentations/Keynote_Maybury.pdf (2012, accessed 28 November 2013).
63. Honeynet. The honeynet project, https://www.honeynet.org/about (March 2009, accessed 12 May 2011).
64. Torrieri D. Cyber Maneuvers and maneuver keys. In: *Proceedings of the IEEE military communications conference (MILCOM 2014)*, Baltimore, MD, 2014.
65. NMAP. Nmap security scanner, https://nmap.org/. (Accessed 12 May 2011).
66. Metasploit. Metasploiit put your defenses to the test, http://www.rapid7.com/products/metasploit/. (Accessed 12 May 2011).
67. Snort. Snort, https://www.snort.org/. (Accessed 6 June 2013).
68. Wikipedia. Zero-day (Computing), https://en.wikipedia.org/wiki/Zero-day_(computing). (Accessed 6 October 2013).
69. US-CERT/NIST Advisory CVE-2014-0160. National Vulnerability Database/NIST, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-0160 (2014, accessed 9 April 2014).
70. Bace R and Mell P. NIST special publication on intrusion detection systems, http://www.21cfrpart11.com/files/library/government/intrusion_detection_systems_0201_draft.pdf (2001, accessed 22 May 2011).
71. Wang K and Stolfo S. Anomalous payload-based network intrsuion detection. In: *Proceedings of the 7th symposium on recent advances in intrusion detection (Springer LNCS 3224)*, Sophia Antipolis, France, 2004.
72. Bolzoni D, Etalle S, Hartel P, et al. POSEIDON: a 2-tier anomaly-based network intrusion detection system. In: *Proceedings of the 4th IEEE international workshop on information assurance (IWIA'06)*, London, 2006.
73. Lala J and Harper R. Architectural principles for safety-critical real-time applications. *Proc IEEE* 1994; 82: 25-40.
74. Heo S, Kim P, Shin Y, et al. A survey on intrusion-tolerant systems. *J Comput Sci Eng* 2013; 7: 242-250.
75. Wang F, Gong F, Sargor C, et al. SITAR: a scalable intrusion tolerance architecture for distributed services. In: *Proceedings of the second IEEE/SMC information assurance workshop*, West Point, NY, 2001.
76. Verissimo P, Neves N, Cachin C, et al. Intrusion-tolerant middleware: the road to automatic security. *IEEE Secur Privacy* 2006; 4: 54-62.
77. Bangalore A and Sood A. Securing web servers using self-cleansing intrusion tolerance (SCIT). In: *Proceedings of the 2nd international conference on dependability*, Athens, Greece, 2009.
78. Nguyen Q and Sood A. A Comparison of intrusion-tolerant architectures. *IEEE Secur Privacy* 2011; 9(4): 24-31.
79. Madan BB, Goseva-Popstajanova K, Vaidyanathan K, et al. A method for modeling and quantifying the security attributes of intrusion tolerant systems. *Perform Eval J* 2004; 56: 167-186.
80. Madan B and Banik M. Attack Tolerant architecture for big data file system. *ACM SIGMERICS Perform Eval Rev* 2014; 41: 65-69.
81. Reed I and Solomon G. Polynomial codes over certain finite fields. *J Soc Ind Appl Math (SIAM)* 1960; 8: 300-304.
82. Stallings W. *Data and computer communications*. 7th ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2007.

Author biographies

Bharat Madan is a professor in the Department of Modeling, Simulation & Visualization Engineering, Old Dominion University, Norfolk, Virginia. Prior to this he was with the Applied Research Lab, Penn State University, where he headed the Distributed Systems Department. He has also held academic positions at the Indian Institute of Technology (New Delhi), Duke University, University of Delaware and the Naval Postgraduate School (Monterey, CA). His teaching and research interests are in computer and network systems security, attack tolerant survivable architectures, real-time systems security, sensor data fusion, and autonomous systems collaboration.

Manoj Banik is a PhD candidate in the department of Modeling, Simulation and Visualization Engineering (MSVE) at Old Dominion University (ODU) in Norfolk, Virginia. He started his PhD program in this department in fall 2013. He received his BS and MS degrees in computer science and engineering from BUET (Bangladesh University of Engineering and Technology) and UIU (United International University), Dhaka and Bangladesh, respectively. He was an assistant professor in the Department of Computer Science at AUST (Ahsanullah University of Science and Technology), Dhaka, Bangladesh from 2006 to 2012. His research interests include computer communications, computer security, algorithms, and multilevel and recurrent neural network.

Doina Bein is an assistant professor in the Computer Science Department at California State University, Fullerton. She has tackled a number of important problems in the area of efficient routing and broadcasting in wireless networks, fault-tolerant coverage (selecting a small subset of nodes to act as communication backbone of a network), and fault-tolerant data structures.