Security Engineering Pilot

Technical Report SERC-2013-TR-036-1

February 28, 2013

Principal Investigator: Dr. Barry Horowitz, University of Virginia

Co-Principal Investigator: Dr. Bill Melvin, Georgia Institute of Technology

Team Members:

University of Virginia: Dr. Peter Beling, Dr. Cark Elks, Dr. Nathan Lau, Dr. Kevin Skadron, Ed Suhler, R.W. Williams

Georgia Institute of Technology: Michael Brinkmann, Dr. Michael Heiges, Jim Perkins, Johanna LoTempio, Tom Owens, Alex Triesczieski

Contract Number: H98230-08-D-0171 Task Order: 0027, WHS, RT 042

Report No. SERC-2013-TR-036-1

The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Assistant Secretary of Defense for Research and Engineering (ASD(R&E)) under Contract H98230-08-D-0171 (Task Order 0027, WHS, RT 042).

Any views, opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense nor ASD(R&E).

ABSTRACT

This report presents the major results for the initial phase of the RT-42 Security Engineering research effort funded through the Systems Engineering Research Center. The work builds upon prior efforts that pointed toward a point defense approach for cyber security that provides defense solutions that are embedded inside of the systems to be protected (as opposed to the access perimeter to those systems and the networks that support those systems). These solutions are referred to as System Aware security because their designs depend upon intimate knowledge of the designs of the systems being protected. The results in the report include:

1) The addition of two new design patterns formatted in the conventional manner used in our earlier efforts. For reasons related to the desire to integrate a design pattern library to promote reuse by others, these new pattern have been integrated into a single section that includes all of the design patterns that have been developed to-date.

2) Design data related to the implementation of a ground-based prototype for a cyber security solution for an autonomous surveillance system onboard an unmanned aerial vehicle. This information will be the basis for the prototype to be developed over the next phase of this project.

3) An updated description of the cyber security architecture decision support tools being utilized to support design decisions for the prototyping project. The tools include support software that continues to be updated as new features are added to the decision support tool set.

Elements of this work have been published in the peer review journal, Systems Engineering, Vol 16, No 3, 2013

TABLE OF CONTENTS

FIGURES AND TABLES

1 System-Aware Design Patterns for Prototype Application of System Aware Cyber Security Surveillance Mission on an Unmanned Air Vehicle

1.1 Introduction

This section of the document describes, in a standardized format, the design patterns to be applied in the Prototype Application of System Aware Cyber Security Surveillance Mission on an Unmanned Air Vehicle project. It includes two design patterns developed under RT-42 and integrates these with previously developed design patterns under RT-28. These design patterns will be combined to provide Parameter Assurance function, the Navigation Assurance and Camera Gimbal Control Assurance functions for the project. Each of these functions can be implemented through the implementations of these design patterns to protect the Piccolo flight control system and the mission surveillance equipment that are part of the Outlaw UAV to be utilized on this project.

1.2 Background

One of the strengths of the perimeter security approach is that it offers a set of standardized commercially available products. In contrast, System-Aware security solutions are highly customized to the applications to which they are embedded. Thus, there is a need to facilitate reuse of System-Aware security solutions across a diverse set of applications. One approach is to create security design patterns. These security patterns could facilitate in the reuse of System-Aware security solutions across additional systems by drawing on the consensus of engineers engaged in building these systems–similar to how they have aided in object-oriented projects [1] and more traditional security technologies [2]. In addition, these patterns would provide documentation characterizing the sufficient conditions for application as well as suggestions for additional synergistic patterns to enable the engineering community to apply them to new and existing systems.

In order to provide a starting point for the exploration and development of new secure design patterns, three patterns are presented based upon the work outlined in this paper. The format for these patterns is based upon those used for traditional perimeter security as presented by Schumacher in his book on “Security Patterns: Integrating Security and Systems Engineering” [2]. However, unlike the patterns presented by Schumacher, these patterns are not based upon implemented solutions but on research cases. Research cases were chosen as, “Patterns support the understanding of problems and their solutions,” [2] and, “Patterns are generic—as independent of or dependent on a particular implementation technology as need be.” [2]. Thus, design patterns provide not only a means for recording implemented solutions, but a method for recording research cases so that they can be applied to problems across a wide set of domains. As System-Aware security aims to provide cyber security solutions that are applicable to many domains, design patterns provide an ideal means of recording and presenting such solutions for reuse.

1.3 Design Patterns

1.3.1 Diverse Redundancy

Name: Diverse Redundancy

Example of Need: Figure 1 presents a high-level system diagram for a typical steam fed nuclear reactor powered turbine control system. As indicated in Figure 1, the turbine receives actuation commands from a controller, currently available from a variety of vendors (e.g., the GE Mark VI, and Triconex Tricon). Operators located in the

main control room of the power plant are responsible for controlling the turbine. These individuals receive status information from the controller that influences their operational actions, which can include stopping the turbine and correspondingly tripping the reactor to stop steam flow into the turbine. In addition to operator actions, the controller receives sensor information (listed in Figure 1) that together influences its automatic control actions. In situations where the turbine operation is such that it is of immediate importance to stop steam flow, the reactor is automatically stopped (i.e. scrammed), with a reactor shutdown process that is supported by the sensor information related to turbine operation.

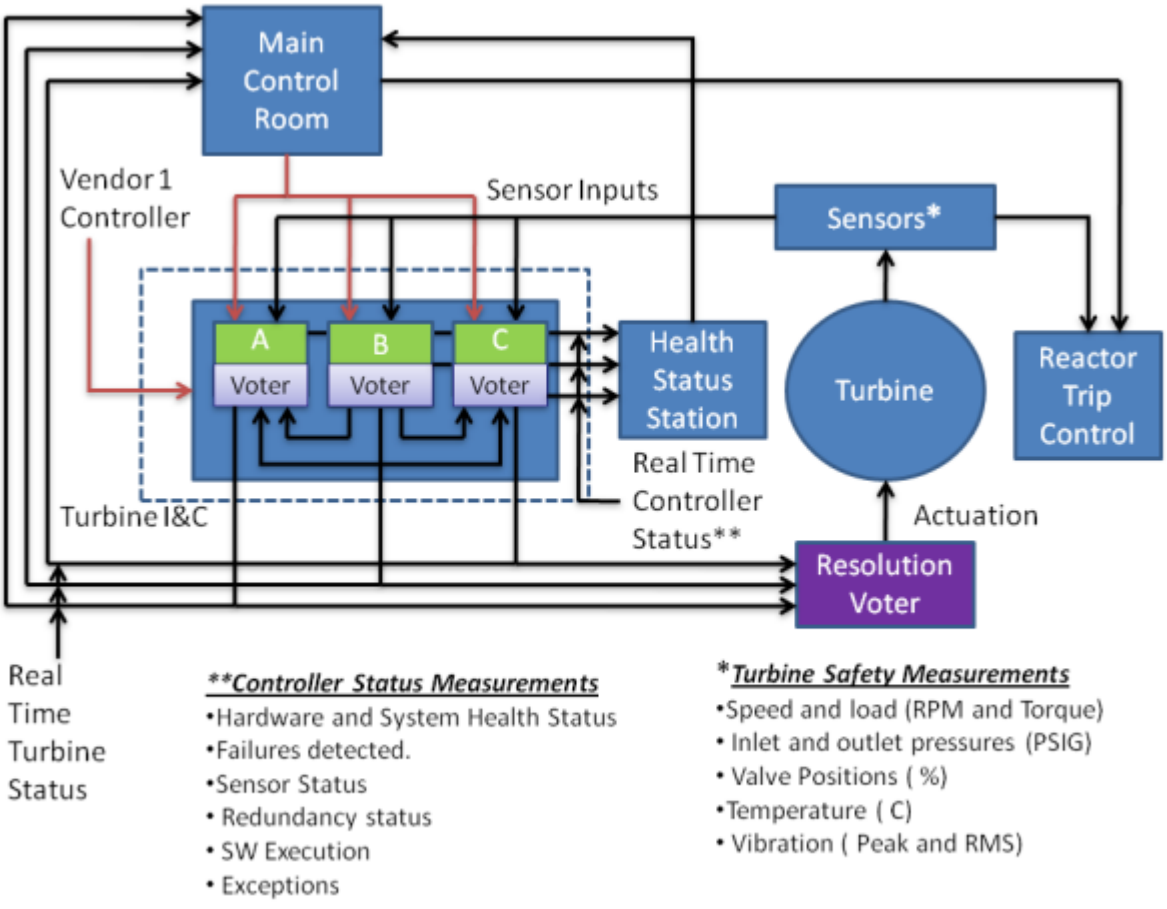


Figure 1: A high-level system diagram for a typical steam fed nuclear reactor powered turbine control system.

The turbine controller is designed to meet high reliability and safety standards by employing redundancy and a resolution voter.

Figure 1 also highlights the fact that nuclear power plant turbine controllers are designed to meet high operational reliability and safety standards, and accordingly often employ various types of redundancy. However, there has recently been a rash of insider attacks where a Trojan horse was found to be embedded into the equipment of the supplier of the reactor's controllers. Given the significant economic consequences resulting from serious damage to the turbine, and the need to shut down (trip) the nuclear reactor in the event of a turbine shut-down, how can the reactor's owner continue to maintain high reliability while ensuring her system against a possible supply chain attack?

Context: Ensure that system functions critical for achieving mission objectives and high reliability requirements will be available even if one or more the components that support those functions have been compromised by a cyber attack.

Problem to be Solved: While the use of redundant components in systems is a common way to assure continuity of operation, the use of components that are susceptible to a common source of failure does not provide assurance against a cyber attack that affects all of the common components.

Solving this problem requires one to resolve the following forces:

- For a cyber attack, a single exploit can be developed and used to compromise all of the identically redundant components that might otherwise provide enhanced continuity of operation.
- The cyber attack can be embedded into the redundant components through the supply chain or an insider attack making it difficult to ensure that a cyber attack has not compromised all of the components.
- The single exploit may be an extremely minor change (e.g. the change of a single parameter) and triggered remotely or based on a certain condition (e.g. time). As a result detecting that a component or components have been compromised can be extremely difficult.

Solution: Solutions for ensuring that the success of a cyber attack on a critical system function(s) does not result in mission failure can be based upon protection approaches developed by the fault tolerant systems community. One such technique is to utilize diversely redundant components to ensure that a system is able to carry out its mission objectives even when one of those components breaks down. This assumes that each of the diversely redundant failures is independent; i.e. no common source exists to cause the same fault in all of the components. A cyber attack is one such common source that could put all redundant components at risk, and prevent a system from completing its mission objectives. This solution mitigates the capacity for a cyber attack to successfully compromise all redundant components by utilizing diverse components with a different set of attributes.

Structure:

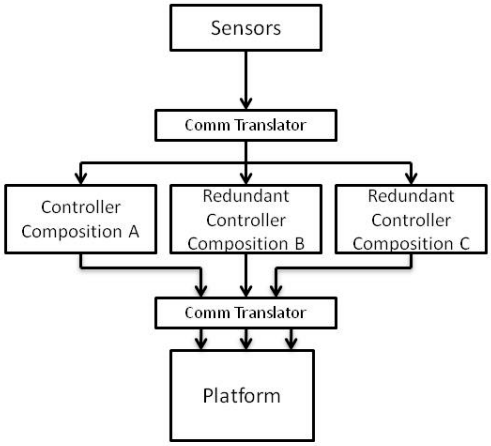


Figure 2: A simple illustration of the structure of Diverse Redundancy.

In this instance three different controllers are used to receive inputs from a set of sensors and issue inputs to control a platform. Furthermore, each of the controllers is utilizes a diverse set of protocols. Thus, communication translators are included (i.e. the Comm Translators).

Diverse Redundancy requires the following elements:

- Two or more diversely redundant components. These components must be diverse with regards to the common source of the cyber attack. For example, if the common source is a Trojan horse injected via the supply chain, then the common source is the supplier and the components should be procured from independent suppliers.
- Special hardware may be needed to integrate the diverse components into the system. For the structure shown in Figure 2, the diverse components use special communication translators as each of the diverse controllers employs a different communication protocol.

Dynamics: As seen in Figure 2, the diverse components will possibly need to be able to receive input, generate output, and exchange information with other diverse components. Depending on whether the original system employed redundancy or not, additional infrastructure may be needed to transmit information to and from the diversely redundant components, as well as between the diversely redundant components. For example, an additional mechanism might need to be integrated into the system which is used to ensure that only one of the diversely redundant controllers is sending its information along and that the remaining are serving as backups. Alternatively, in order to avoid bumpy outputs when it is required to switch components due to a failure, a mechanism could be employed to average the outputs of the diversely redundant components. This result is then utilized as the output of the diversely integrated components.

Implementation: Diversity can encompass a large set of parameters, including hardware, software, vendor, geographical location, administrator(s), etc. Thus, it is important to consider the type(s) of diversity that will be needed to prevent an attack. For example, utilizing multiple diverse operating systems will force an adversary to develop cyber attacks for each of the operating systems, but could leave them vulnerable to an attack embedded in a common hardware component. Diverse components may require special hardware and/or software to ensure interoperability.

Example of Need Resolved: The owner of the nuclear reactor decides to integrate two additional turbine controllers along with *Verifiable Voting* and *Physical Configuration Hopping* (see Figure 3). As the reactor owner was worried about compromised components originating from the supplier, she has decided to integrate three turbine controllers from different vendors. As each of these vendors employs its own communication protocol, additional communication translators are needed to ensure interoperability. *Verifiable Voting* has been utilized to detect and isolate a controller issuing potentially damaging information, as well as to ensure that only one of the controller's command signal reaches the turbine. Finally, *Physical Configuration Hopping* is utilized to both enhance security and select which of the diversely redundant controller's data will be passed to the turbine and which are serving as backups.

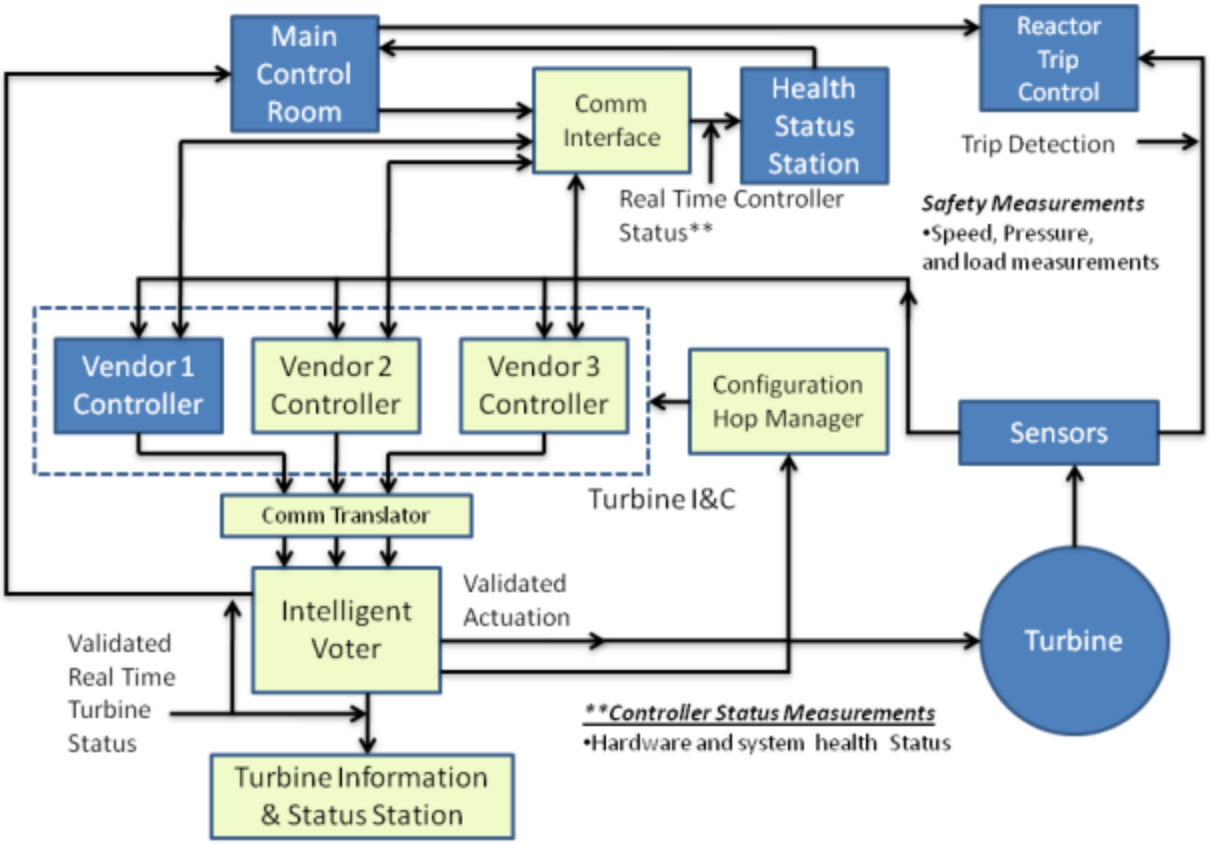


Figure 3: Resolved solution for *Diverse Redundancy*.

In this instance, diver redundancy and *Verifiable Voting* have been employed to protect the turbine controller and ensure protection against a supply chain attack.

Variants: A variation includes utilizing redundant components that possess reduced or different capabilities. For example, a GPS-based navigation system can utilize an inertial navigation system as a redundant backup.

Known Uses: [3, 4, 5]

Consequences: The following benefits may be expected from applying this pattern:

- *Diverse Redundancy* can serve to increase the complexity of an attack that would attempt to compromise all components by forcing the need for cyber attacks with specific capabilities to address each of the diversely redundant components
- In systems without redundant components, *Diverse Redundancy* can potentially increase the systems robustness to faults
- Some systems may already possess diverse components and can possibly make implementation easier

The following potential liabilities may arise from this pattern:

- *Diverse Redundancy* may require additional infrastructure to ensure interoperability with all components

- In systems without redundant components, *Diverse Redundancy* may require new infrastructure to ensure all components receive the appropriate input and that the proper output signals are sent
- As *Diverse Redundancy* requires the components to be diverse with regards to the common source of failure, the amount of commercial off the shelf (COTS) solutions for providing diversity may be limited
- Life cycle costs and training of support staff could increase due to the requirement to service diversely redundant components

Related Design Patterns: *Verifiable Voting* is a mechanism that can be combined with *Diverse Redundancy* to help detect and isolate which of the diversely redundant components have been compromised. *Diverse Redundancy* can also be combined with *Physical or Virtual Configuration Hopping* to dynamically switch which component is engaged in the operational system at any given time in order to both detect a compromised component and minimize the time available for an exploit to affect the system.

1.3.2 VERIFIABLE VOTING

Name: Verifiable Voting

Example of Need: A museum has recently installed a video surveillance system to protect its collection of rare and valuable artifacts. As shown in Figure 4, this system consists of a series of security cameras that transmit their data to a media server and its hot shadowed backup. Security personnel can pull the video streams from the media server to their mobile devices to observe the rooms remotely. In addition, when the museum is closed, the media servers scan all of the incoming video streams for unauthorized personnel. If the servers detect any unauthorized access an alert is sent to the security personnel. The security personnel can then decide to pull the video stream to determine the situation and take appropriate action to apprehend the intruder. Recently the primary employee responsible for managing and maintaining the media servers was fired under the suspicion that she was planning a heist on the museum. Given the access this employee was afforded to the media servers, the owner of the museum is concerned that the employee may have already tampered with the media servers as part of the planned heist. As a result, the museum owner wishes to employ additional security to protect against a possibly malicious server.

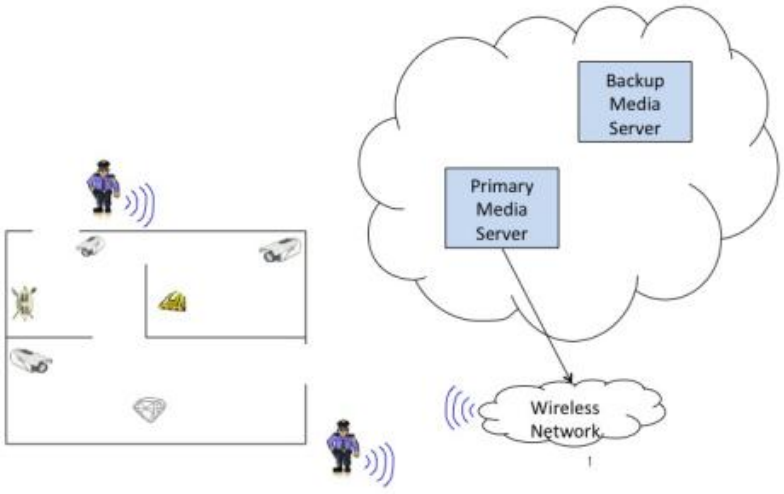


Figure 4: A high-level system diagram of a video surveillance system for a museum.

The security cameras send the video surveillance to media servers that distribute the information wireless to security personnel.

Context: Systems often produce information that is critical in determining the appropriate set of actions to be taken to ensure the desired outcome. However, this can potentially result in a significant decline in system performance when there is reason to suspect that the source of information may not always be producing reliable information. This decline can potentially lead to undesired or inferior outcomes whenever the source is producing valid information, but nonetheless is not trusted, or the source is trusted, but producing bad information—such as due to a cyber attack. Thus, a method is needed to be able to detect and/or isolate those components that may be compromised and may be producing faulty information.

Problem to be Solved: How can one continue to utilize (i.e. trust) the outcomes of a critical system when one suspects that the system has been compromised?

Solving this problem requires one to resolve the following forces:

- If the system were compromised by a cyber attack, it could cause considerable damage. However, simply disabling the system is undesirable as the support it affords is critical to achieving the desired outcomes. Thus, a method is needed to detect when the output of the system is valid and when it is misleading.
- It may be possible to restore a system to working order once a compromise has been detected; however, to do so it is may be necessary to isolate the component responsible for producing the faulty output
- To protect against a cyber attack, the mechanism employed to detect and isolate systems producing faulty information must also be secured. In addition this mechanism must not impact system performance to the point of preventing the system from functioning properly.

Solution: A voting scheme is typically used to detect and isolate systems that are producing faulty outputs. Voting can also be utilized to detect misleading outputs. However, if the misleading information is being produced as a result of a cyber attack, it is possible that the attack may have been embedded into the component through the supply chain or from an insider. As a result, it is possible that the mechanism used to carry out the voting may be compromised. *Verifiable Voting* is utilized to provide voting in a secure manner. It is based on providing a hierarchy of voters tailored to the specific needs of the system to ensure that components acting maliciously are identified, while not significantly impacting system performance. Each of the voters in the hierarchy is designed based upon a trade-off analyses regarding ease of verifiability—i.e. confidence that it has not be compromised—and ability to perform timely and complex comparisons.

Structure: *Verifiable Voting* is composed of one or more voting mechanisms (e.g. the Byzantine fault tolerant voter [6] or Civitas [7]) implemented in hardware or software. This includes an extremely simple voting mechanism, implemented in hardware or software, which is easily verifiable; i.e. known to be secure. However, such a simple mechanism may only be capable of implementing a simple voting scheme. This may result in voting rules that do not include all available information, resulting in an unacceptable degradation of performance compared to a voting scheme that uses more information. Alternatively, using more information may make the voting logic too complex to sufficiently verify its implementation from a security standpoint. As a result, in addition to using the less sophisticated, but more verifiable voters to validate simple, but mission critical machine generated outputs (e.g. fire the gun), they can also be used periodically, as a coarse check on whether a less verifiable voter has been compromised. Finally, *Verifiable Voting* requires that there be multiple redundant systems producing output. The amount of redundancy determines how many of the redundant systems can be compromised before it becomes impossible to detect and isolate potentially compromised components. Figure 5 illustrates one possible hierarchy of voters that assumes only a single redundant system will be compromised at a given moment.

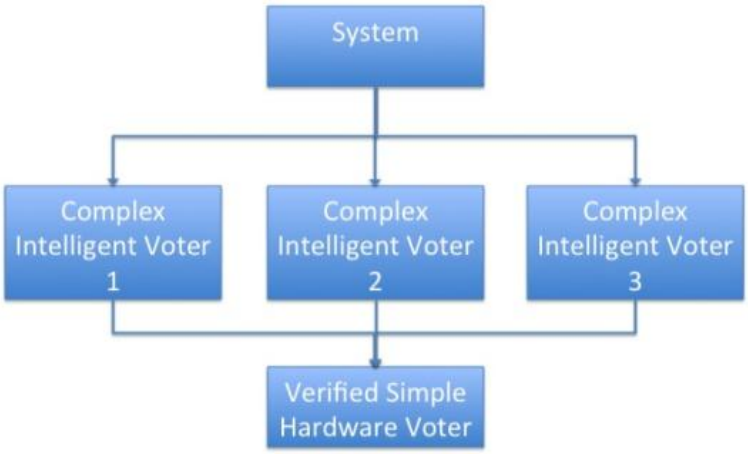


Figure 5: A simple example of *Verifiable Voting*.

This includes three complex intelligent voters that are used to evaluate the information from the system. These results are fed to a simple hardware voter that can be easily verified.

Dynamics: All voters need to be able to receive the necessary outputs for comparison from the multiple redundant systems. It is important that the most verifiable (i.e. secure) of the hierarchy of voters be able to override the decisions of the less secure voters.

Verifiable Voting requires replication of the outputs of the system in order to carry out the vote. If the system already carries the necessary redundancy or the output of the system is small (e.g. a true or false value) then the cost of this replication can be negligible. However, when the outputs being voted on are large (e.g. the output of diversely redundant video streams received over a wireless network for voting) then such voting can add significant overhead. While, this overhead can potentially be mitigated through the use of additional resources, it may also be possible to mitigate it through the use of customized system designs. For example, in Figure 5 each of the three complex intelligent voters is receiving the three inputs simultaneously. However, it is possible to stagger the voting across each voter; i.e. complex intelligent voter 1 receives the three inputs and votes, than complex intelligent voter 2 receives the three inputs and votes, and finally complex intelligent voter 3 receives the three inputs and votes. Once this is done each of the complex intelligent voters can send its simplified results to the simple hardware voter for a final decision (see Figure 5). For the case of a wireless network communicating the information, this scheme of staggered voting can result in a reduction in bandwidth utilization, while also potentially delaying the detection of any modification of data in one of the streams.

Implementation: When implementing *Verifiable Voting* it necessary to determine an appropriate scheme for voting as well as the input that will be voted on. Given this information, it is possible to determine the desired number of redundant system components to achieve detection and isolation. It is also possible to develop an appropriate hierarchy of voters. This hierarchy will depend on the type of information used in voting, the frequency of voting, and the desired security of the *Verifiable Voting* scheme itself. Finally, additional resources or techniques may be needed to ensure that the desired level of system performance is achieved.

Example of Need Resolved: To defend the museums rare artifacts against a possible cyber attack embedded in the media server, the owner decides to implement *Verifiable Voting*. As there are only two media servers, *Verifiable Voting* is only able to provide detection. As the museum has security guards on patrol and possesses the capacity to rapidly lock down the artifacts, it is decided that isolation is not necessary. If the *Verifiable Voter*

detects a problem (i.e. cannot reach consensus) it will alert the security personnel who can then place the museum on lockdown.

To ensure that the *Verifiable Voter* will be secured against cyber attacks, it is decided that the *Verifiable Voter* will be deployed onto mobile devices used by the security personnel for alerts. While it is possible that a single guard's device could be compromised, there would still be several additional security guards capable of receiving the information. Thus, an attacker would have to compromise all of the mobile devices used by personnel. From the perspective of the museum owner, this is deemed an unlikely event and thus an acceptable risk.

Variants: None.

Known Uses: [3, 4, 5]

Consequences: The following benefits may be expected from applying this pattern:

- Can both detect misleading output as well as isolate the offending component
- Voting mechanism can be implemented in a more secure manner
- Offers a flexible implementation to trade off desired level of security with cost, complexity, and performance impacts

The following potential liabilities may arise from this pattern:

- Detection and isolation require the introduction of multiple redundant components with the attendant liabilities (see the design pattern for *Diverse Redundancy* in section 1.3.4)
- Depending on the information being voted upon, it can result in an increase in complexity and cost to ensure that solution meets the desired goal
- Can be defeated if enough of the redundant devices are compromised to form a majority (what constitutes a majority will depend on the voting scheme utilized)

Related Patterns: This pattern can be combined with *Diverse Redundancy* to potentially increase the difficulty in compromising all redundant components—e.g. through an insider or supply chain attack.

1.3.3 PHYSICAL CONFIGURATION HOPPING

Name: Physical Configuration Hopping

Example of Need: Modern ships are equipped with a wide set of systems to monitor and control (e.g. engine, propulsion, fire suppression, and climate control). A company wishes to produce a lower cost ship by consolidating the network between the monitoring consoles and the physical systems into a single COTS network switch. To improve the reliability of the design, a redundant network switch is installed to resume operations in the event the primary switch fails. However, consolidating all network connections also leaves the entire ship vulnerable to any cyber attacks embedded into the primary network switch:

- Send potentially misleading information to the monitoring systems
- Could disable the ship through a denial of service attack by dropping all communications
- Modify or inject commands to the physical systems in order to damage, disable or misdirect the ship

Context: Ensure that critical system components that have been infected with a cyber attack will be unable to actively disrupt, damage, or misdirect systems operations.

Problem to be Solved: Techniques exist to detect, isolate, and disable system components that are behaving in a manner to cause harm to the system. However, a system component compromised by a cyber attack has the potential to disrupt and possibly damage critical system components before such methods are successfully able to disable the offending component. In addition, such methods may be unable to prevent cyber attacks aimed at passive monitoring or more sophisticated attacks that attempt to cause disruptions and damage more subtly (e.g. Stuxnet attack).

Solving this problem requires one to resolve the following forces:

- Ensure that a cyber attack is not given enough time to cause damage or disrupt system operations; this time may be less than the time needed to detect and isolate the compromised component
- Prevent a cyber attack exploit from reading enough information to form a coherent data set for use by the attacker
- Security solution must not compromise the systems mission objectives by significantly impacting on system performance

Solution: Solutions for preventing compromised system components from taking potentially malicious action can be based on techniques developed by the cyber security community. One such technique is moving target defense; a technique that aims to dynamically switch functionality across multiple resources. *Physical Configuration Hopping* builds on this technique by continuously shifting control between multiple redundant physical system components in order to disrupt a cyber attack before it can cause permanent damage.

Structure: As seen in Figure 6, *Physical Configuration Hopping* requires multiple redundant components to be dynamically interchanged (two in Figure 6). This dynamic reconfiguration determines which component(s) is in control at any given time. In addition, there is a mechanism to the control the frequency of the dynamic readjustment as well as determine which component is in control—in Figure 6 it is the configuration hop manager. Finally, their needs to be a mechanism in place to control the switching between components; this includes the frequency of hopping, as well as the order of hopping from one component to another (pertinent to cases of higher orders of redundancy).

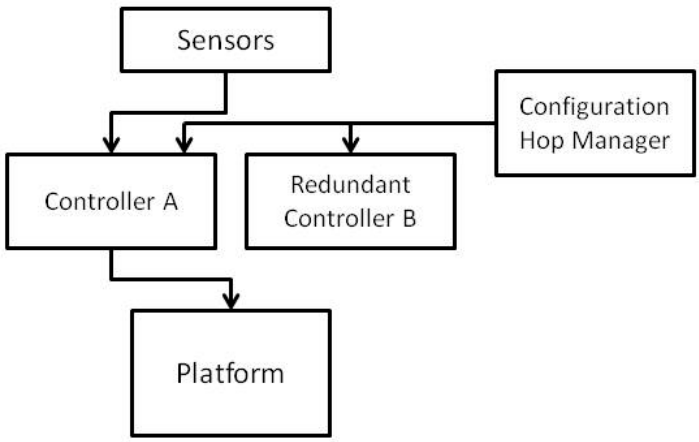


Figure 6: A simple Physical Configuration Hopping setup.

This instance includes dynamic reconfiguration across two redundant controllers. Controller A is currently set to the active controller.

Dynamics: *Physical Configuration Hopping* requires that all redundant components be able to receive and generate output to the appropriate systems, as control will need to be dynamically switched between those components. In addition, it may be necessary to ensure that the dynamic switching between components is bumpless. For example at the time of switching the multiple redundant components may be in different states; thus, the switch between components results in an unintended switching of states.

Implementation: When implementing *Physical Configuration Hopping* it is important to consider the time it will take for a compromised component to cause damage. For example, a turbine in a nuclear reactor can potentially be damaged in a matter of seconds. Alternatively, it may take several minutes or even hours to steer a ship far enough off course to be considered damaging. In addition, the sophistication involved in switching between redundant system components depends on the sophistication of the cyber attack to be prevented. For example, switching between redundant components in a round robin fashion may disrupt a cyber attack that is just trying to transmit damaging commands quickly. However, a more sophisticated attack may be able to detect the switching patterns. This information could then potentially be used to issue commands that ultimately cause damage through controlled thrashing that occurs every time a switch from the compromised component to a non-compromised component occurs. It is also important to decide how much control is given to administrators to change the frequency of hopping as well as alter the algorithm used to control the switching order and specific, perhaps pseudo-randomized, timing.

Example of Need Resolved: The ship building company decides to combine *Physical Configuration Hopping* with *Diverse Redundancy* in order to protect the ship from a compromised network switch. The company decides to purchase two switches from different vendors in order to help prevent a scenario where both switches are compromised via the supply chain. The company then determines that it is not worried about a Trojan horse being embedded in the new system component used for monitoring the information, as control and status information between systems is not of direct value to an attacker; however, it is worried about a compromised switch causing denial of service or injecting false and/or damaging commands. It is then determined that it would take at least five minutes before a compromised network switch could cause any permanently damaging actions. Finally, the dynamic switching has the potential to cause some status information to be lost; however, the amount of information lost is small relative to the frequency of updates; i.e. no additional resources are needed for bumpless control.

Variants: *Virtual Configuration Hopping*

Known Uses: [3, 4, 5]

Consequences: The following benefits may be expected from applying this pattern:

- Prevent a system component compromised by a cyber attack from being able to compromise the mission objectives; prevention can occur independently, and faster than methods used for detection, isolation, and restoration
- Makes the development of cyber attacks more difficult by introducing time as an element

The following potential liabilities may arise from this pattern:

- Requires multiple redundant components with the attendant liabilities of the *Diverse Redundancy* design pattern
- Introduce the need for methods to ensure bumpless control
- Defeated if the frequency of hopping is too slow, or the algorithm for switching is predictable

Related Patterns: Can be combined with *Diverse Redundancy* to potentially mitigate the risk that multiple redundant components will be compromised.

1.3.4 VIRTUAL CONFIGURATION HOPPING

Name: Virtual Configuration Hopping

Example of Need: An e-commerce business stores customer credit card information in a secure facility equipped with a video surveillance system. This video surveillance is maintained and routinely inspected by a private contractor to ensure that it is operating properly. Recently the company has learned that several of the companies that also use this private contractor have been the victims of theft. An investigation of each of the sites has revealed that each of the systems responsible for receiving and displaying the streams to security personnel was infected with a Trojan horse to perform a simple replay attack. Furthermore, it is suspected that an employee of the private contractor did the theft. The e-commerce site has invested significant resources in building the secure facility as well as the video surveillance system and desires a solution to secure the video surveillance system against a possible insider attack.

Context: Ensure that critical system functions that have been infected with a cyber attack will be unable to actively disrupt, damage, or misdirect systems operations.

Problem to be Solved: Techniques exist to detect, isolate, and disable system functions that are behaving in a manner to cause harm to the system. However, a system function compromised by a cyber attack has the potential to disrupt and possibly damage critical system functions before such methods are successfully able to disable the offending functions. In addition, such methods may be unable to prevent cyber attacks aimed at passive monitoring or more sophisticated attacks that attempt to cause disruptions and damage more subtly (e.g. Stuxnet attack).

Solving this problem requires one to resolve the following forces:

- Ensure that a cyber attack is not given enough time to cause damage or disrupt system operations; the time to cause damage or disruption may be less than the time needed to detect and isolate the compromised function
- Prevent a cyber attack exploit from reading enough information to form a data set for use by the cyber attack
- Security solution must not compromise the systems mission objectives by significantly impacting system performance parameters

Solution: Solutions for preventing compromised system functions from taking potentially malicious action can be based on the techniques developed by the cyber security community. One such technique is moving target defense that aims to dynamically switch functionality among multiple resources. *Virtual Configuration Hopping* builds on this technique by continuously shifting control between multiple redundant virtualized system functions in order to disrupt a cyber attack before it can cause permanent damage.

Structure: As seen in Figure 7, *Virtual Configuration Hopping* requires multiple redundant functions to be dynamically interchanged (two in Figure 7). This dynamic reconfiguration determines which function(s) is in control at any given time. In addition, there is a mechanism utilized to the control the frequency and exact timing of the dynamic readjustment as well as determine which function is in control—in Figure 7 it is the configuration hop manager. Finally, their needs to be a mechanism in place to control the switching between function; this includes the frequency of hopping, as well as the order of hopping from one function to another (pertinent to cases of higher orders of redundancy).

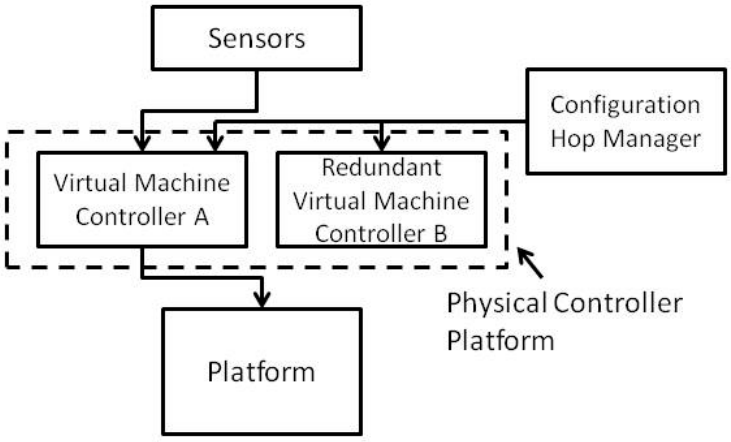


Figure 7: A simple *Virtual Configuration Hopping* setup.

This instance includes dynamic reconfiguration across two virtually redundant controllers located on the same physical platform. Controller A is currently set to the active controller.

Dynamics: *Virtual Configuration Hopping* requires that all redundant functions will be able to receive and generate output to the appropriate systems, as control will need to be dynamically switched between those functions. In addition, it may be necessary to ensure that the dynamic switching between functions is bumpless. For example at the time of switching the multiple redundant functions may be in different states; thus, the switch between functions results in an unintended switching of states.

Implementation: When implementing *Virtual Configuration Hopping* it is important to consider the time it will take for a compromised function to cause damage. For example, a turbine in a nuclear reactor can potentially be damaged in a matter of seconds. Alternatively, it may take several minutes or even hours to steer a ship far enough off course to be considered damaging. In addition, the sophistication involved in switching between redundant system functions depends on the sophistication of the cyber attack to be prevented. For example, switching between redundant functions in a round robin fashion may disrupt a cyber attack that is just trying to transmit damaging commands quickly. However, a more sophisticated attack may be able to detect the switching patterns. This information could then potentially be used to issue commands that ultimately cause damage through controlled thrashing that occurs every time a switch from the compromised function to a non-compromised component occurs. It is also important to decide how much control is given to administrators to change the frequency of hopping as well as alter the algorithm used to control the switching order and specific, perhaps pseudo-randomized, timing.

Example of Need Resolved: The concerned e-commerce business determines that the system responsible for receiving and displaying information can be virtualized quickly at minimal costs and decides to use *Virtual Configuration Hopping*. The e-commerce site sets up a virtualized environment to run multiple copies of the system. In addition, the e-commerce site obtains a video surveillance application from another vendor and adds that into its virtual environment. Once this has been set up, the e-commerce business determines that it should be concerned regarding the possibility of the credit card information stored at the protected site being stolen. It then determines that it would take an intruder at least 10 minutes to download all of the credit card information. The system is then set-up to hop between the virtualized system functions every 5 minutes. However, during switching the video feed appears to exhibit some slight distortions (i.e. it is bumby). To mitigate this effect, *Virtual Configuration Hopping* system is updated to provide a smooth (i.e. bumpless) stream.

Variants: *Physical Configuration Hopping*.

Known Uses: [3, 4, 5]

Consequences: The following benefits may be expected from applying this pattern:

- Prevent a system component compromised by a cyber attack from being able to compromise the mission objectives; prevention can occur independently, and faster than methods used for detection, isolation, and restoration
- Makes the development of cyber attacks more difficult by introducing time as an element

The following potential liabilities may arise from this pattern:

- Requires multiple redundant functions with the attendant liabilities of the *Diverse Redundancy* design pattern
- Introduce the need for methods to ensure bumpless control
- Defeated if the frequency of hopping is too slow, or the algorithm for switching is predictable

Related Patterns: Can be combined with *Diverse Redundancy* to potentially mitigate the risk that multiple redundant functions will be compromised.

1.3.5 DATA CONSISTENCY CHECKING USING DIVERSE STATE ESTIMATIONS

Name: Data Consistency Using Diverse State Estimations

Example of Need: The operation of a turbine involves the transfer and display of data to allow an operator to monitor the output. A main control room exists where an operator views the current state of a turbine, information is sent from sensors so that the operator can track the turbine's movement and other variables (speed, temperature, etc.). Operators observe the output to determine if any variables exceed a given threshold, at which time they are expected to take action to stop the turbine or sound an alarm. However, if a cyber attack corrupts the outputs of the sensors or the displayed data, an operator could be made to think that the operation is "as normal"; when in reality an attack is underway. This situation illuminates the need for a security solution that data displayed to operators is correct and not a misrepresentation resulting from a cyber attack?
Context: Ensure that system data presented to operators for use in system control can be trusted and has not been altered by a cyber attack.

Problem to be Solved: Perimeter solutions can prevent a large percentage of attacks, but experience has shown that perimeter solutions are not sufficient. For example attackers have found ways to bypass perimeter security solutions through insider and supply chain attacks. One class of attacks is the case where an element of the system is compromised and the normal monitoring function of the operators is simultaneously corrupted to mask the attack.

Problem to be Solved: Perimeter solutions can prevent a large percentage of attacks, but experience has shown that perimeter solutions are not sufficient. For example attackers have found ways to bypass perimeter security solutions through insider and supply chain attacks. One class of attacks is the case where an element of the system is compromised and the normal monitoring function of the operators is simultaneously corrupted to mask the attack.

Solving this problem requires one to resolve the following forces:

- An exploit that changes the state of the physical machinery and disguises that from the operator
- An exploit that changes the data display to influence an operator into thinking an attack is underway
- Supply chain attacks embedded in electronics
- Insider inserted attacks

Solution: This design pattern utilizes diversely derived state estimations to verify the integrity of the data shown to the operator. The state of a system can be estimated through use of different state-related measurements; for example, speed can be derived from position, temperature is related to speed, and other general relationships exist between different states of a system. These relationships between states can be represented in discrete time mathematical equations that represent the interaction among states of a system as a function of time. By calculating the system state using measurements that do not directly provide the outputs shown to the operator for control purposes, the data integrity can be checked before being displayed. This potentially can alert an operator to a hidden attack, or a feigned attack that does not actually require system shutdown. This solution prevents cyber attacks from compromising data by using state estimation.

Structure:

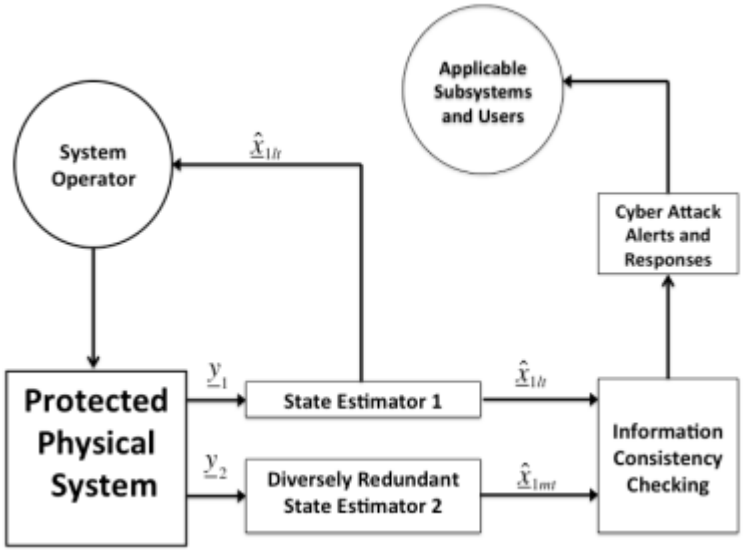


Figure 8: A simplified diagram for the data integrity detection system.

State estimation requires the following elements:

- Measurements, taken from turbine sensors that may not be currently factored into the operator display, but can indirectly be used to estimate the displayed states for confirmation purposes.
- An alarm or detection warning of some form to alert the operator that the output can no longer be trusted, or a control action that reconfigures the system to employ an alternate, diverse path for display of information

Dynamics: During an exploit of this form, an attacker is changing the presentation to the operator display continuously. It is also possible that the attacker is changing the feedback control system of the turbine to create continuously changed performance of the physical element (in this case, turbine). Therefore the attacker is using

the system itself to dynamically change what the operator sees to push them into taking action when none is needed, or taking no action when an attack that impacts system performance is actually happening**.** An effective solution must address the dynamics posed by the attackers efforts.

Implementation: In order to avoid false alarms, rigid testing of the system to be estimated must be undertaken. In certain scenarios, false alarms or missed alarms could have varying costs, making one of more importance than the other. Wherever the system is being implemented, the specific situation must be analyzed. Similarly, it is important to determine how much time the system can withstand being under attack before an alarm must be called. In other words, the longer that measurements can be gathered, the more confidence one can have in calling an alarm on a cyber attack. However, in cases where the system will rapidly deteriorate from an attack, this must be accounted for.

Other necessary analysis revolves around sensor accuracy, alarm threshold, and the window size it demands. Window size is the number of measurements that are considered at once to determine if an attack is occurring. The threshold is the point at which a measurement is considered unusual- usually determined by a set number of standard deviations away from the mean. A sliding window scale can be utilized to set a limit at the number of points in one time window that can be above the threshold, and if that is exceeded an alarm is called. A larger window size allows more confidence in the alarm but also may result in further degradation the system. Also, utilizing a low threshold will increase the probability of false alarms, but will decrease the probability for missed detections. All of these "settings" of the integrity checker must be determined based on the normal performance of the underlying system and the predicted nature of attacks on the system.

Example of Need Resolved: An owner of a turbine decides to implement diverse state estimation. An insider supply chain attack occurs, where the attacker makes an attempt to misrepresent the turbine. The operator is given information that the turbine is experiencing high temperature levels, which would typically result in a shutdown. However, the state estimation data integrity checker uses other measurements to be confident that the turbine temperature has remained within normal tolerances, and the perceived change is the result of a cyber attack. The operator is alerted and no action is taken, avoiding a costly shutdown and alerting the company to the attempted cyber attack.

Variants:

- Sensors selected for integrity measurements
- The specific diverse redundancy techniques used for estimation
- The recipient of the alarm or alert
- Method of detection: specific techniques for data analysis that can be used
- Detection technique, sensors, information distribution
- Security techniques used to protect the data collection and analysis process

Known Uses: [8]

Consequences: The following benefits may be expected from applying this pattern:

- *State Estimation* can serve to increase the complexity of an attack that would be necessary to successfully mask a systems true state
- Operators will be able to detect intruders quickly and accurately
- Detection will then allow the avoidance of large costs from unnecessary shutdowns or damage caused by a fail to shutdown

The following potential liabilities may arise from this pattern:

- False alarms: if a turbine is shut down incorrectly, this could result in high costs to owners
- Multiple false alarms over a short period of time- could have negative regulatory or government action

Related Design Patterns: *State estimation* can be combined with diverse redundancy to provide restoral mechanisms. For example, if there are two channels for information distribution to the operator, the channel can be switched to the secondary, still trusted channel after an attack.

1.3.6 SYSTEM PARAMETER ASSURANCE

Name: System Parameter Assurance

Example of Need: The military is currently hunting for the location of an enemy base that has been serving as a base of operations for planning and operations of a string of bombing attacks against civilian targets. To facilitate in this search effort for this facility, a new set of Unmanned Aerial Vehicles (UAVs) have been outfitted and deployed with advanced surveillance equipment. However, it has recently been discovered that key personnel responsible for assembling these scouting and surveillance UAVs was sympathetic to the enemy's cause and may have installed equipment with embedded trojan horse. Specifically there is concern that this trojan horse may tamper with one of the numerous parameters used to drive the navigation and surveillance equipment of the UAVs in order to prevent them from detecting the enemy facility. For example, there is concern that the trojan horse will, during the course of a mission, change the designated flight plan of the aircraft to prevent it from locating and taking pictures of the enemy facility. In addition, as such an embedded trojan horse would require only modifying a small set of parameters—potentially only a single parameter—on board the aircraft, it would require a substantial amount of time to validate every single piece of hardware and every line of code to determine if a UAV had been compromised. Furthermore, as such a modification would be extremely difficult for an operator to detect, if such an attack occurred it would likely not be detected. Due to the importance of the UAVs to finding the enemy facility, a solution is needed to protect the UAV's parameter information.
Context: As a rule, systems include parameter tables to allow for ease of control and configuration changes. These tables provide the opportunity for attackers to gain control of how the system operates.

Problem to be Solved: Many systems utilize parameter tables to control how the system functions. Such tables allow the system to be more rapidly and easily reconfigured to suite a wide variety of situations. However, such tables provide adversaries with a potentially easily exploitable target that can be used to prevent a system from being able to fulfill its mission objectives (e.g. Stuxnet attack).

Solving this problem requires:

- Monitoring system parameter tables to determine when critical parameters have been changed
- The capability to distinguish parameter changes that are legitimate—i.e. those changes that are the result of an action taken by an operator acting in good faith or the system behaving to specification—from those that are fraudulent—i.e. those changes that are the result of a malicious cyber attack
- Determining the appropriate actions to take once a fraudulent process for restoring parameters

Solution: The solution utilizes an on-board monitoring capability to regularly monitor the parameter values in use and to compare these to pre-stored values or authorized changes. If differences are observed, checks are made to determine the source and validity of the change (e.g., use key logging HW/SW to monitor the inputs made at an operator control station to trace back the change to a valid action taken by an operator). In the case where no valid source for a parameter change can be detected, the solution treats this as an attack with the need for resolution. One such resolution would be to restore the parameter and inform the appropriate system operator.

Structure:

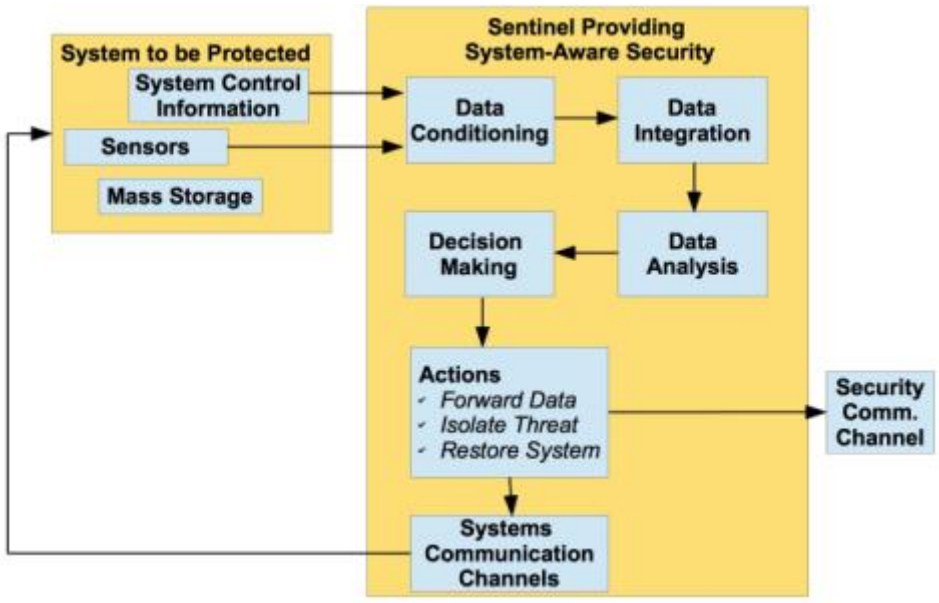


Figure 9: An example of the *System Parameter Assurance*.

In this example, the design pattern is realized as a separate Sentinel Providing System-Aware Security (i.e. *System Parameter Assurance*) component, processing the necessary HW/SW functionality needed to protect and restore the critical parameters used to control the system to be protected.

System Parameter Assurance requires the following elements:

- Data Conditioning function to prepare the collected parameter data for comparison to pre-stored values in the Sentinel, as well as prepare the expected potential legitimate sources of the changes for evaluation
- Data Integration function for sending the collected data to the appropriate computing element for data analysis
- Data Analysis function for determining if a parameter change was the result of a legitimate action or the result of action taken by a malicious cyber attack
- Decision Making function to determine the course of action upon discovery of an illegitimate parameter change (e.g. restore a parameter's value, communicate changes to the appropriate operator
- Components for taking Actions, including changing the parameter tables, communicating the information to the appropriate operators and establishing an alternative, secure mechanism for future mission changes to the parameters
- Security Communications Channel to provide a secure mechanism for exchanging necessary information regarding parameter changes

Dynamics: The rate at which parameters are monitors must conform to the dynamics of the parts of the system that those parameters impact. For example, strategic parameters can be monitored less frequently whereas dynamic control parameters must be monitored more frequently.

Implementation: The degree to which parameter changes are accomplished automatically or semi-automatically depends upon the readiness to trust machine restoration decisions and the urgency of the change. In addition, supplementary actions may be required based on inferences of the intent of the change and mission to be accomplished. This requires the solution to incorporate sufficient functionality to enable interaction with

appropriate system operators in order to take advantage of the operator's judgment about the mission and the threats to the mission. Certain parameter changes may be easily traced to legitimate sources, while others may not. This could also impact the role of the operator in the detection and response loop, as well as, the necessary support structure. The creation of a specialized, secure method of communication used only for attack responses may be needed to provide the necessary mechanism for disallowing future unauthorized changes to parameters.

Example of Need Resolved: In order to ensure that an embedded trojan horse will not be able to disrupt the UAV's from successfully completing their mission, a light weight Sentinel providing *System Parameter Assurance* is integrated into the all potentially compromised UAVs to protect their critical system parameters from fraudulent changes. If the Sentinel detects a fraudulent change, then it will automatically restore system parameters to the previous value. In addition, in order to help identify which UAVs have been compromised, the Sentinel will report all fraudulent changes to appropriate operators. During the course of the next scouting mission, as the UAV approaches the enemy facility an embedded trojan horse activates to try and change the UAV's course. However, the Sentinel detects the unauthorized change and immediately puts the UAV back on course. The Sentinel also informs the mission commander of the violation. The UAV is able to successfully locate the enemy facility. Upon the UAV's mission completion, the commander orders a full inspection to be made in order to remove the faulty components.

Variants: None

Known Uses: Ongoing DOD research activity with the Steven Institute's System Engineering Research Center.

Consequences: The following benefits may be expected from applying this pattern:

- Parameter monitoring should mitigate asymmetric attacks that can exploit the simplification offered through system parameter table control

The following potential liabilities may arise from this pattern:

- Could redirect more sophisticated attackers to embedded software solutions that override the existing parameters during program execution.
- The monitoring function will need to be securely protected against exploits that would be directed at its parameter assurance functions.

Related Patterns: None

1.4 WORKS CITED

[1] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Reading, MA: Addison-Wesley, 1995.

[2] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann and P. Sommerlad, Security Patterns: Integrating Security and Systems Engineering, Hoboken, NJ: John Wiley & Sons, 2006.

[3] R. A. Jones and B. Horowitz, "System-Aware Cyber Security Architecture," *Systems Engineering,* vol. 15, no. 2, pp. 224-240, 2012.

[4] R. A. Jones, T. Nguyen and B. Horowitz, "System-Aware Security for Nuclear Power Systems," Boston, MA, 2011

[5] G. Babineau, R. A. Jones and B. Horowitz, "A System-Aware Cyber Security Method for Shipboard Control Systems with a Method Described to Evaluate Cyber Security Solutions," Boston, MA, 2012.

[6] L. Lamport, R. Shostak and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems,* vol. 4, no. 3, pp. 382-401, 1982.

[7] M. Clarkson, S. Chong and A. Myers, "Civitas: Toward a Secure Voting System," 2008.

[8] B. Horowitz and K. Pierce, " Application of Dynamic System Models and State Estimation Technology to the Cyber of Physical Systems," *Systems Engineering,* 2012.

2 Implementation Data Package for Implementing Design Patterns and Emulating System Performance for the Prototype Application of System-Aware Cyber Security Surveillance Mission on an Unmanned Aerial Vehicle Projects

2.1 Introduction

This section of the document " System-Aware Design Patterns for Prototype Application of System Aware Cyber Security Surveillance Mission on an Unmanned Air Vehicle Prototype Application of System-Aware Cyber Security Surveillance Mission on an Unmanned Aerial Vehicle Projects," details the security design patterns and provides the general description for the security functions that will be implemented in this project. For this project, the critical system functions to be secured reside in the Piccolo II flight and sensor control system of the surveillance subsystem. This section of the document provides more detailed information regarding the interfaces between the security solutions and the Piccolo and surveillance system, as well as the implementation of the prototype Sentinel for system monitoring, attack detection, and system restoration. It is intended that the same interfaces can be utilized for both the Piccolo and the sensor subsystems. In addition, this section of the document provides design information regarding the ground-based emulation capability that will be utilized for design support and solution evaluation, prior to design of a flight capable Sentinel.

2.2 Sentinel / Piccolo II Interface Design

Figure 10 presents a top-level view of the integration environment for protecting the Piccolo II flight control system using a Sentinel employing the intended design patterns presented in, "System-Aware Design Patterns for Prototype Application of System Aware Cyber Security Surveillance Mission on an Unmanned Air Vehicle."

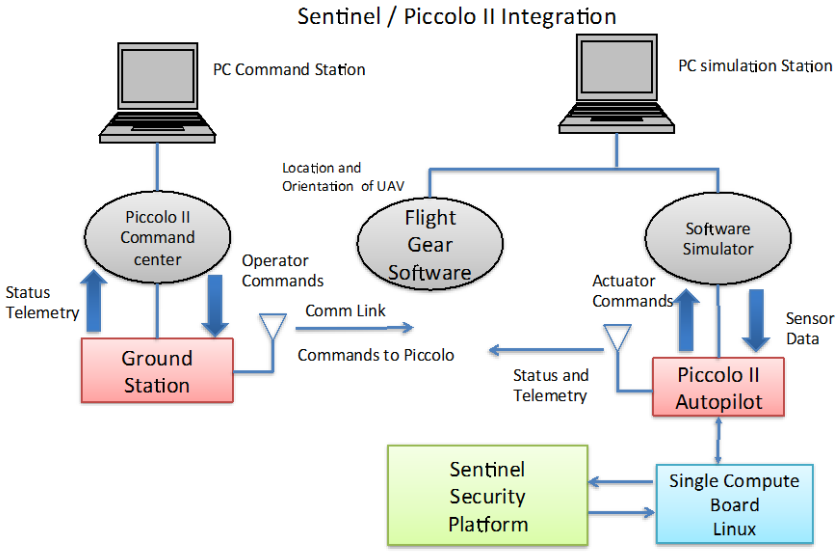


Figure 10: A top-level view of the integration environment for protecting the Piccolo II Autopilot system.

Protection of the system is accomplished through a Sentinel Security Platform that is able to monitor and restore the Piccolo II Autopilot via the interfaces provided by a Single-Compute Board.

In section 2.3, we discuss our decision to utilize CloudShield, a commercial network-monitoring Sentinel, as the Sentinel Security Platform for the initial ground-based prototype. In section 2.4, we discuss the selection our intended single-board compute platform, Phidgets, as the interface for connecting the CloudShield and the Piccolo II.

A more detailed assessment of the Piccolo's design reveals the use of RS-232 (i.e. Serial) protocols for connecting to external devices. A Serial interface is not viewed as best serving the implementation needs to support the usage of our design patterns. As a result, conversions from RS-232 to packet-based communication over Ethernet emerge as a design requirement.

Figure 11 illustrates the hardware configuration for the Piccolo. The prototype Sentinel will accomplish its monitoring functions by accessing information through the MPC555 processing board. This will include access to system parameters, navigation information, and mission hardware configuration data. In addition, example exploits representing supply chain based attacks will be implemented on the MPC555 processor. We have only recently received delivery of the Piccolo II hardware and complementary ground equipment and software; as a result, we are currently in the process of validating our ability to accomplish the needed interfaces.

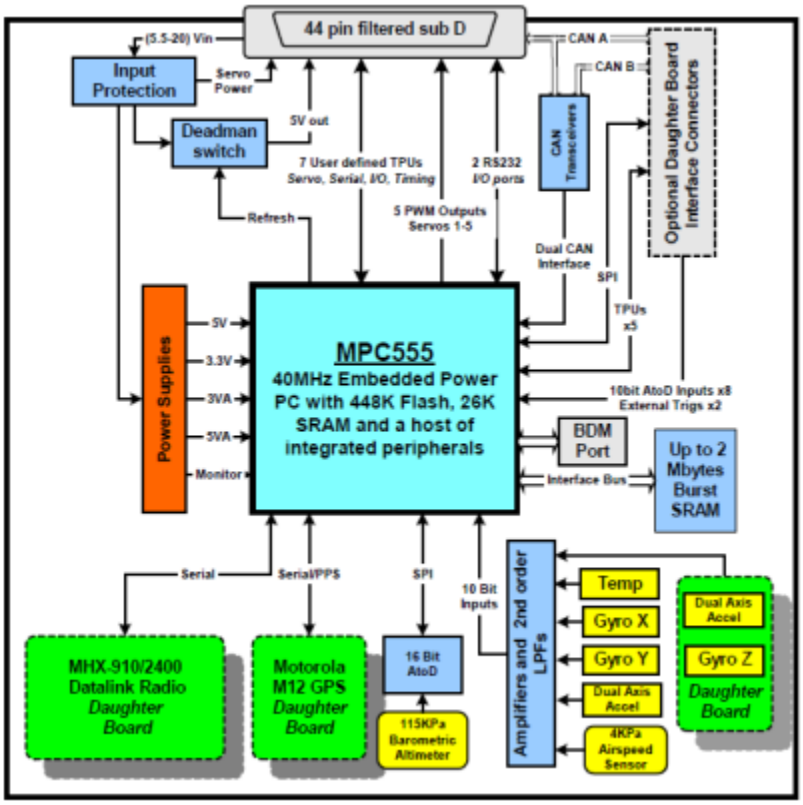


Figure 11: Hardware configuration for the Piccolo II flight control system.

In addition, the Piccolo II includes a supporting simulation environment that, for our purposes, permits ground-based evaluations while emulating an aircraft in flight. This feature of the Piccolo II system has already been put to use, allowing initial explorations of the *System Parameter Assurance* design pattern. For example, we have demonstrated that we can access information on waypoints, flight control parameters, and user passwords. In addition, we will be able demonstrate exploits, as well as employ the intended design patterns to protect against those exploits. Using the simulation environment, we have conducted early demonstrations of the ability to track operator inputs into ground station devices. We can also monitor communications to the Piccolo to confirm

external parameter control inputs. Taken together, this early work provides confidence in the ability to transfer the necessary technology into the ground emulation environment, including the live Piccolo II system. With respect to exploits, the simulation capability has allowed us to demonstrate the ability to emulate a supply chain attack that changes parameters. During the time required to substitute the live Piccolo system for the simulator, we will continue to carry out development activities using the simulation environment.

2.3 CLOUDSHIELD

The prototype system will employ an off-the-shelf network security product called CloudShield as the Sentinel Security Platform shown in **Error! Reference source not found.**0. While CloudShield includes many of the desirable features of a programmable Sentinel, it does not meet the size, weight, and power requirements for airborne use. As a result, we will then transfer the designs developed using the ground-based prototype to a Sentinel configured for flight in a later phase of this work. This permits the design team to better separate the design topics of cyber security effectiveness and the footprint requirements for flight by first developing effective algorithms and then converting the software to operate on new, flight-capable hardware.

Assessment and initial use of the CloudShield network monitor has already occurred and supported the prototype development approach for the *System Parameter Assurance* and *Data Consistency* design patterns. As software is designed for CloudShield, an alternate shadow hardware configuration will be considered as the potential airborne version. As the effort progresses, more specific decisions can be made about the final hardware configuration.

The Sentinel design concept derived from earlier work includes developing security features for the Sentinel itself. Two of these features will be capability to perform HW/SW configuration hopping and software fingerprinting by measuring machine utilization. CloudShield includes a redundant processing module that will allow us to explore the usage of the *Physical Configuration Hopping* design pattern as a method for protecting the Sentinel itself from attack.

An assessment has been conducted to relate CloudShield features with those features that will be needed on the flight-capable Sentinel. Figure 12 presents the results of that assessment.

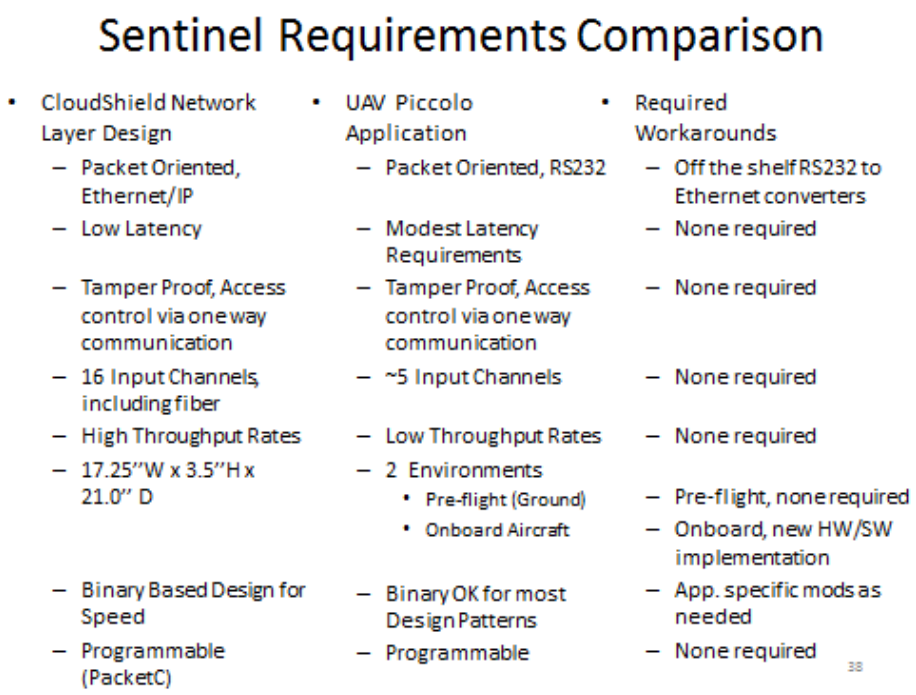


Figure 12: Table detailing the results of an assessment to relate CloudShield features to those needed for a flight-capable Sentinel.

Left column details the CloudShield features. Middle column the needed flight-capable features. Right column presents workarounds to interface the CloudShield with the Piccolo II.

2.4 SINGLE-COMPUTE BOARD: PHIDGETS

As shown in Figure 10, the single-compute board serves as the interface between the MPC555 in the Piccolo II and the CloudShield based Sentinel. As indicated in section 2.1, conversions are needed between the native RS-232 interfaces on the Piccolo and the packet-based communications we need over Ethernet. These conversions will be accomplished using available off the shelf converters that can be interfaced with a variety of off the shelf single-computing boards. Our intention is to use the widely employed Phidgets board based on its ability to host a variety of operating systems, large number of interfaces to connect to external devices, and monitoring sensors (e.g. temperature and proximity) that provide additional opportunities for recognizing attacks. In addition, the Phidgets board is a candidate for use on board the flight ready version of the Sentinel. Regarding operating system selection for the Phidgets, current candidates include Debian based Linux and Windows. Finally, diversely redundant design patterns can be supported by the existing Phidgets HW/SW structure.

3 DECISION SUPPORT DATA PACKAGE

3.1 BACKGROUND

An outcome of RT-28 was the creation of a framework for selection and integration of design patterns that together provide the basis for selecting the most desirable security architecture for a given system. This framework requires several teams with diverse orientations to derive the desired architecture:

- Blue Team to select the most critical functions to protect
- Red Team to identify potential asymmetric attacks (i.e. attacks that are low in complexity but high in impact) impacting the critical system functions and to assess the impact of the potential design patterns on complicating their attack vectors
- Green Team to recognize cost constraints in the selection of design patterns for application

This effort, RT-42, developed an initial decision support tool to support the teams engaged in selecting the desired architecture. Figure 13 presents a flowchart representing the activities required of the three teams charged with deriving the desired architecture. The RT-42 computer based support tools are Microsoft Excel based and provide a vehicle for the teams to sequentially explore the design space, ultimately leading to a desired architecture.

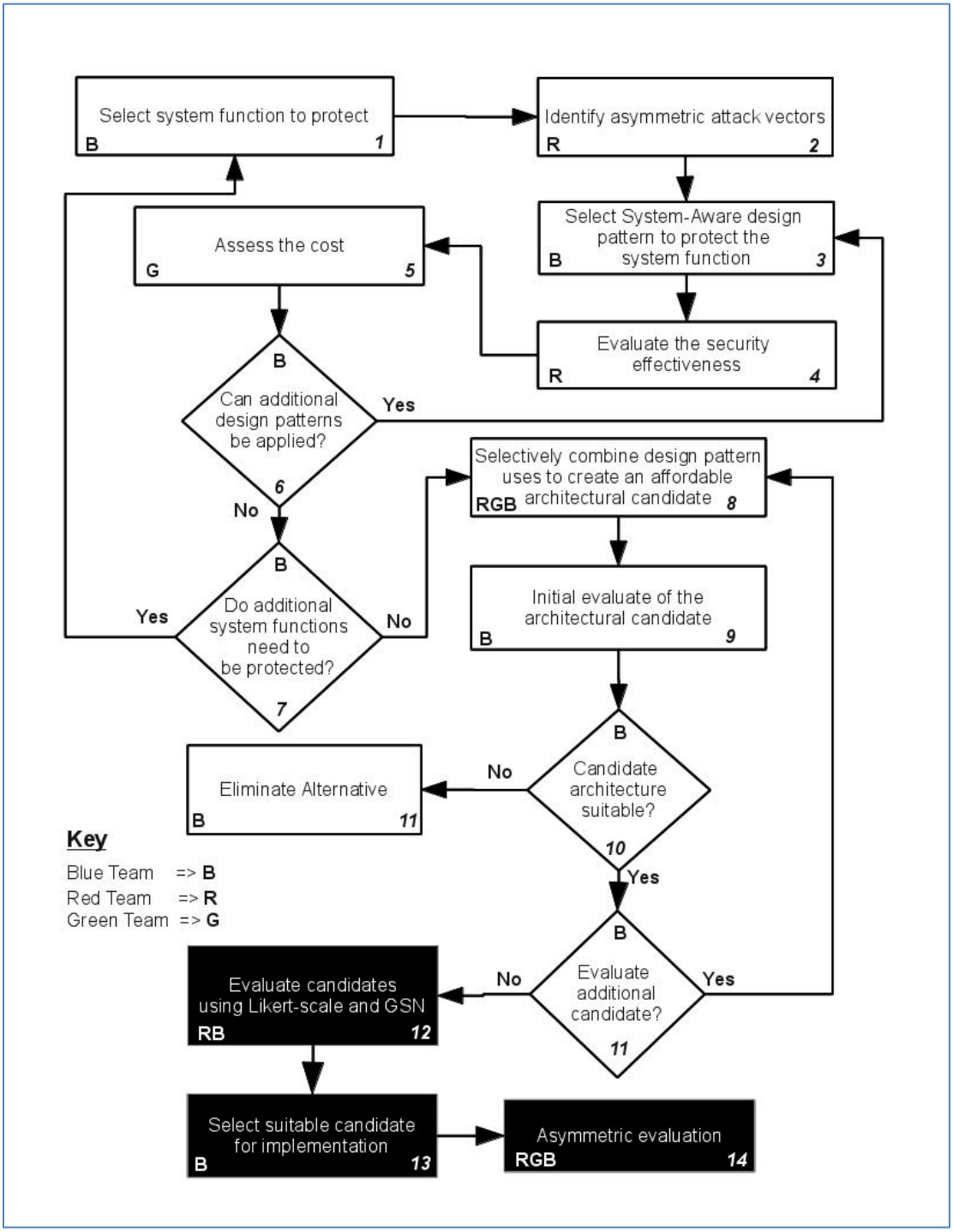


Figure 13: Activities for Deriving Desired Architecture

3.2 Tool Description

This section of the document there were two principle goals in developing this prototype system. First, was to develop algorithms for automatically generating architectural candidates. Second, was the creation of software tools to help users explore the set of all candidate architectures by making adjustments to the automatically generated architectures. This also included the creation of tools and visualization aids to help users compare and contrast the candidate architectures. Finally, while these make up the main contribution of the prototype decision support system, it is noted that the prototype is fully functional; i.e., it supports all of the steps laid out in Figure 13.

3.2.1 Selection of System Functions for Protection

Figure 14 shows how the prototype system supports the selection of system functions for protection for a simple example system. As can be seen, three system functions have been identified and assigned a relative ranking according to their importance to protect by the Blue Team. For the system shown in Figure 14, the System Control function has been designated the most important system function to protect and the User Display the least; i.e., a higher ranked function is more important than a lower ranked function. It is noted that while only three system functions are shown here, the prototype system can support any number of functions.

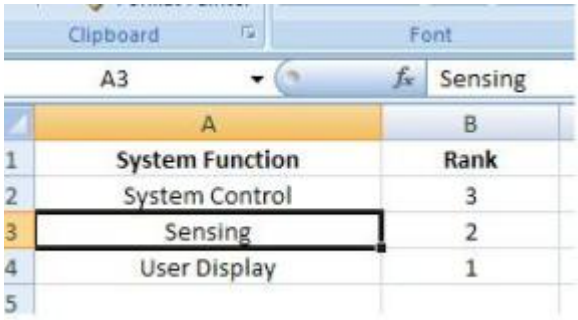


Figure 14: Represents the prototype decision support system's support for selecting system functions to protect and assigning them a relative rank ordering.

3.2.2 Selection of System-Aware Design Patterns

After the system functions that would benefit from System-Aware security solutions have been selected, System-Aware design patterns are chosen by the Blue Team to protect these functions. Figure 15 shows this for the System Functions identified in Figure 14. As can be seen, the user first selects one of the identified system functions from a drop down list. After selecting a function for security consideration, a System-Aware design pattern is chosen for that system function. Finally, each combination of system function and System-Aware design pattern is assigned a unique identifier.

The prototype system offers two methods for creating this identifier. The method illustrated in Figure 15 uses the previously designated relative rank combined with a short acronym of the System-Aware design pattern. This method is intended to produce an identifier that would allow the user to intuitively recognize the relative importance of the system function as well as the method used to protect it. However, this scheme requires all available System-Aware design patterns to be assigned a unique short hand identifier before the process is started. Depending on the number of design patterns available, this may not be practical. To address this potential pitfall, a second method is available. This method recognizes that a System-Aware design pattern can only be applied to a system function once; i.e., each combination of system function and design pattern should be unique. A unique identifier is generated by hashing these unique combinations. This has the benefit of generating

unique identifiers without requiring additional information from the user; however, unlike the former method, the identifiers will not have intuitively derived meaning.

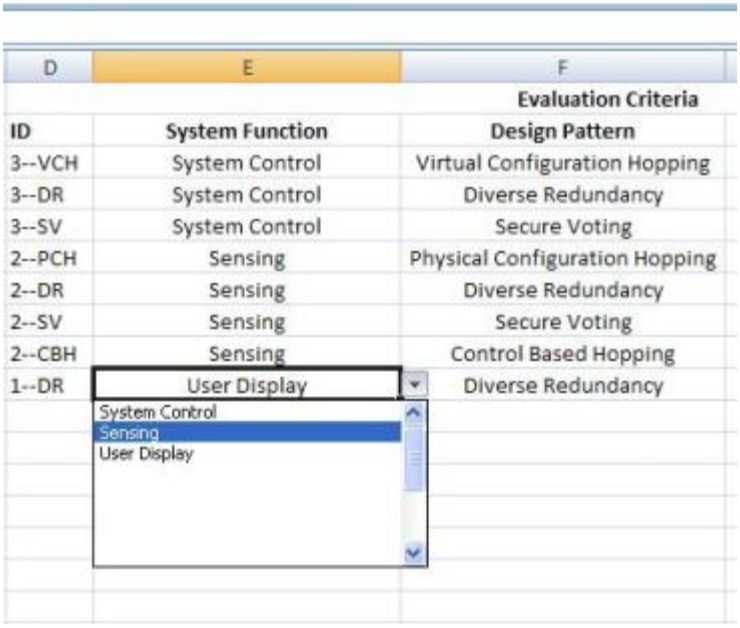


Figure 15: Illustrates how the prototype system is used to support the selection of System-Aware design patterns.

For this instance the available system functions are assumed to be those presented in Figure 14. As can be seen, only those functions previously selected for protection can be selected in this step. Also note that every combination of system function and System-Aware design pattern is assigned a unique identifier. For this instance, the identifier is composed a combination of the relative rank of the system function and the first letter of every word of the System-Aware design pattern.

3.2.3 Determining the Resources Necessary to Develop the System-Aware Architecture

Every combination of system function and System-Aware design pattern should be evaluated independently to determine its costs and collateral impacts on the system. For the prototype architecture, it is assumed that all resources can be represented by a single monetary cost. This is more restrictive a system where there can be multiple costs represented in different units (e.g., money, power, and space). The decision to reduce this to a single monetary cost was made to both support a more robust set of algorithms for automatically generating architectural candidates, and to make it easier for a user to compare and contrast competing architectural candidates.

Figure 16 builds upon Figure 15 to illustrate how this is done in the prototype system. As can be seen, every combination of system function and System-Aware design pattern is assigned a single monetary cost. For the prototype system, these costs are considered to be independent.

D	E	F	G
		Evaluation Criteria	
ID	System Function	Design Pattern	*Cost*
3--VCH	System Control	Virtual Configuration Hopping	$3,000.00
3--DR	System Control	Diverse Redundancy	$2,000.00
3--SV	System Control	Secure Voting	$2,500.00
2--PCH	Sensing	Physical Configuration Hopping	$3,000.00
2--DR	Sensing	Diverse Redundancy	$4,900.00
2--SV	Sensing	Secure Voting	$1,150.00
2--CBH	Sensing	Control Based Hopping	$2,400.00
1--DR	User Display	Diverse Redundancy	$1,000.00

Figure 16: Illustrates how the prototype system supports the assessment of the necessary resources needed to implement a given System-Aware architecture.

In this instance every combination of System-Aware design pattern and system function generated in Figure 15 is assigned a monetary cost.

3.2.4 Determining the Security Effectiveness

Figure 16 illustrates how the prototype system supports the evaluation of the security potentially afforded by each combination of system function and System-Aware design pattern to be evaluated. In the prototype evaluation system the ID, System Function, and Design Patterns, fields are automatically populated based on the inputs received in the previous steps The Deterrence Score is used to represent the security effectiveness potentially afforded by each combination of system function and design pattern. The security effectiveness score can be any integer value; however, for the prototype system, this score is limited to 1, 2, 3, or 4, with higher scores relating to more value. This limitation is not imposed by the evaluation system, but rather stems from the criteria used to determine the effective security score

- Score = 4, Complexity, cost, and time to develop exploits are high and the probability of a successful exploit is low
- Score = 3, Complexity, cost, time to develop exploits are high and the probability of a success exploit is high
- Score = 2, Complexity, cost, time to develop exploits are low and the probability of a success exploit is low
- Score = 1, Complexity, cost, time to develop exploits are low and the probability of a success exploit is high

The particular system in 17 is built using those combinations illustrated in Figure 13.

3.2.5 Selecting Architectural Candidates

The selection of architectural candidates is performed through a combination of automation tools and user input. The user provides constraints that are used to guide the automated execution of algorithms in the creation of a subset of candidate System-Aware architectures. In addition, the user is then able to modify one or more of these candidate architectures to generate additional candidate architectures. This process can result in one of two outcomes

1. A set of candidate architectures to be evaluated using a more rigorous criteria
2. Select the architecture that will be implemented

The prototype system can support both outcomes; however, the prototype only supports the creation of candidate architectures and does not provide any additional support to perform a more rigorous evaluation.

3.2.5.1 User Constraints

The prototype decision support system accepts user constraints on the total amount of resources that are available for implementing System-Aware design patterns to protect system functions. Since the prototype system assumes that only one monetary resource exist, this step is presumed to be the user allocating a budget. Figure 16 illustrates the options available to the user (it assumes the resources shown in Figure 15).

- User is allowed to set a value between 0 and the sum total of the resource costs of selecting every combination of system function and design pattern (this is 19,950 in 7)
- User can move a slider between the values of 0 and the sum total of the resource costs of selecting every combination of system function and design pattern (this is 19,950 in 7)

The maximum budget value is automatically computed based on the inputs supplied. Furthermore, the budget values are always integer—the maximum value is always rounded up to ensure that is can cover the difference. This is done to support the automatic generation of candidate architectures.

Figure 17: Illustration of the budget slider used in the prototype system.

This slider is currently set to a budget of $10,000, has a maximum possible value of $19,950, and a minimum value of $0.

3.2.5.2 Automated Support to Generate Candidate Architectures

After a maximum budget has been set and System-Aware design patterns have been selected, their security effectiveness evaluated, and their cost determined for each of the system functions identified as possibly benefiting from System-Aware security, an automated decision support system is then utilized to generate two exemplar System-Aware architectures

1. *Blue Perspective* – The prototype decision support system will select a system function to protect based upon the importance placed on protecting that system function by the system design team. First, the prototype system will select the highest ranked (i.e., most important) system function. Next, System-Aware design patterns will be selected to maximize the protection of the chosen system function. Design patterns will be selected until either the costs meet the available budget or all available patterns have been selected. In the former case, design patterns will be selected to maximize the total security effectiveness offered (the sum of the selected combinations of system functions and design patterns) within the user defined budget. This process will be repeated until either all of the identified system functions have been protected by all available System-Aware design patterns or the total cost meets the maximum user defined budget. It is assumed that the costs and security effectiveness scores for each

combination are independent of all other combinations; thus, the total cost and security effectiveness of the candidate architectures can be computed through a simple summation of the individual values.

2. *Red Perspective* – The prototype decision support system will select system functions and corresponding System-Aware design patterns in order to maximize the security effectiveness of the final System-Aware architecture. To do so the prototype decision support system makes the same assumptions as the *Blue Perspective:* that the individually assigned scores for security effectiveness and the costs can be computed from a simple summation in order to derive the values for the candidate architecture. In the event that two or more System-Aware design patterns consume the same amount of resources and afford the same security effectiveness for different system functions, and only one of those functions can be protected, the importance placed upon protecting the system function by the system design team will be used to determine which of the functions is protected.

For both of these cases, it is recognized that the maximization of the security effectiveness subject to a budget constraint is an instance of the knapsack problem. As a result, the prototype decision support system has been designed to try and take advantage of this fact. First, it is known that the knapsack problem can be solved in pseudo polynomial if all of the weights (i.e., resource costs associated with each combination of system function and design pattern) are non-negative integers. As discussed earlier, the weights are monetary cost estimates. This means that all of the weights are nonnegative. In addition, the maximum budget is an integer value. Finally, the prototype system will round up all of the individual weights to integer values before trying to maximize the security effectiveness. This last step ensures that all of the weights (i.e., costs) are integers. This last step is deemed acceptable as the costs of each of the proposed solutions (i.e., combinations of system functions and design patterns) are considered to be large enough that the change can be safely ignored. Of course, it is still possible that the time required to determine the optimal solution is unacceptable. As a result, the prototype systems allows user to use a heuristic algorithm in place of the optimal solution. This algorithm is not guaranteed to find the optimal solution, but it will run in polynomial time.

These candidate architectures are meant to represent two edge cases, thus providing a starting point for user the exploration of the available design

- *Blue Perspective* – Protects the system by protecting system functions according to the system design teams evaluations
- *Red Perspective* – Protects the system by maximizing the security effectiveness of the final architecture based upon the evaluations of the cyber-attack assessment team

3.2.6 TOOLS TO SUPPORT USER EXPLORATION

After the prototype decision support system has generated the two candidate architectures, the user is then able to adjust those architectures to generate additional candidates. The prototype decision support system offers several tools to support the user in this given task

- Overview and summary statistics describing the candidate architecture generated using the *Blue Perspective* approach. As seen in Figure18, this includes a list detailing which combinations of system functions and design patterns were selected and summary information about this architecture
    - Deterrence Score: The sum of the deterrence scores of the selected combinations as well as the deterrence score of summing all possible combinations
    - Cost: The sum of the costs of the selected combinations as well as the costs of selecting all combinations

  - Higher Ranked Blue: Represents the number of important system functions that were included in the candidate architecture. A system function is important if its relative ranking is greater than or equal to the maximum ranked function divided by half
  - Lower Ranked Blue: Represents the number of less critical system functions that were included in the candidate architecture. A system function is less critical if its relative ranking is less than half the maximum ranked function
  - Bigger Det Red: Represents the number of design patterns included in the candidate architecture that contributed a significant amount of effective security. A significant amount of effective security is a security effectiveness score greater than or equal to half the maximum possible score. For the prototype decision support system this is a score of 4 or 3
  - Smaller Det Red: Represents the number of design patterns included in the candidate architecture that contributed a small amount to the effective security. A small amount of effective security is a security effectiveness score less than half the maximum possible score. For the prototype decision support system this is a score of 2 or 1
- Overview and summary statistics describing the candidate architecture generated using the *Red Perspective* approach. As seen in, this includes a list detailing which combinations of system functions and design patterns were selected and summary information about this architecture. This information is exactly same as that discussed earlier for the *Blue Perspective*
- Two quad charts representing the summary statistics of the candidate architectures. The first chart, seen in Figure 20, shows all of the combinations including in the candidate architectures (*Blue Perspective* and *Red Perspective*). This information includes the relative importance of the system function being protected and its contribution to the effective security of the candidate architecture. This also includes the amount the combination contributes to the overall costs of architecture (this is represented by the size of the glyph). The second chart, seen in Figure 21, shows the same information as the first; however, the combinations plotted are those NOT included in the candidate architecture
- The user should be able to create additional architectural candidates by adjusting those generated through more automated means. In the prototype architecture this is done by allowing the user to select a candidate architecture and add or remove combinations of system functions and design patterns. This interface is shown in Figure 22. In addition, the prototype decision support system highlights how these changes affect the selected architecture. These include highlighting which combinations have been added and removed, as well as highlighting any positive or negative changes in the summary statistics. An example of this can be seen in Figure 23
- When the user creates a new architectural candidate, the prototype decision support system allows the user to save that candidate so it can be compared to all other candidate architectures created. The prototype system provides tools to allow for the user to compare candidates in terms of their security effectiveness, costs, and selected combinations of system functions and design patterns in a pair wise manner
- The prototype decision support allows the user to keep a history of all candidate architectures created, the steps (i.e., adjustments to the automatically generated architectures) that were taken to create those candidates, and a history of the reasoning behind those changes

	Deterrence Score (Blue)	Cost (Blue)				
	14 -- 23	9650 -- 19950				
Higher Ranked Blue	Lower Ranked Blue	Bigger Det Red	Smaller Det Red			
4	1	3	2			
Not Included	Included	Swaped Out	Swaped In			
		Hierarchical Goal Ranking (Blue)				
ID	System Function	Design Pattern	*Rank*	*Deterrence Score*	*Cost*	*Value Factors*
3--VCH	System Control	Virtual Configuration Hopping	3	3	3000	Deterrence; Restoration
3--DR	System Control	Diverse Redundancy	3	3	2000	Restoration
3--SV	System Control	Secure Voting	3	2	2500	Deflection; Restoration
2--PCH	Sensing	Physical Configuration Hopping	2	4	3000	Deterrence; Restoration
2--DR	Sensing	Diverse Redundancy	2	3	4900	Restoration
2--SV	Sensing	Secure Voting	2	2	1150	Deflection; Restoration
2--CBH	Sensing	Control Based Hopping	2	2	2400	Deterrence; Restoration
1--DR	User Display	Diverse Redundancy	1	4	1000	Deterrence

Figure 18: The architecture candidate created automatically using the *Blue Perspective*.

The top displays summary information, including deterrence score, total costs, and rough break down of the importance of the system functions protected and the effectiveness of the design patterns chosen to protect those functions. The bottom shows which combination of system functions and design patterns were selected. This instance was generated using the information shown in Figure 15 and Figure 16.

		Deterrence Score (Red)	Cost (Red)			
		15 -- 23	9650 -- 19950			
	Higher Ranked Blue	Lower Ranked Blue	Bigger Det Red	Smaller Det Red		
	4	1	3	2		
		Maximize Deterrence Architecture (Red)				
ID	System Function	*Design Pattern*	*Rank*	*Deterrence Score*	*Cost*	*Value Factors*
3--VCH	System Control	Virtual Configuration Hopping	3	3	3000	Deterrence; Restoration
3--DR	System Control	Diverse Redundancy	3	3	2000	Restoration
3--SV	System Control	Secure Voting	3	2	2500	Deflection; Restoration
2--PCH	Sensing	Physical Configuration Hopping	2	4	3000	Deterrence; Restoration
2--DR	Sensing	Diverse Redundancy	2	3	4900	Restoration
2--SV	Sensing	Secure Voting	2	2	1150	Deflection; Restoration
2--CBH	Sensing	Control Based Hopping	2	2	2400	Deterrence; Restoration
1--DR	User Display	Diverse Redundancy	1	4	1000	Deterrence

Figure 19: The architecture candidate created automatically using the *Red Perspective*.

The top displays summary information, including deterrence score, total costs, and rough break down of the importance of the system functions protected and the effectiveness of the design patterns chosen to protect those functions. The bottom shows which combination of system functions and design patterns were selected. This instance was generated using the information shown in Figure 15 and Figure 16.

Figure 20: Quad chart displaying all of the selected combinations of system functions and design patterns in the candidate architectures generated by the prototype decision support system for the *Blue Perspective* and the *Red Perspective*.

Those combinations selected as part of the *Blue Perspective* are represented by blue diamonds. Those combinations selected as part of the *Red Perspective* are represented by red circles. The size of the glyph (diamond or circle) represents that combinations relative contribution to the overall cost (bigger glyph represents a larger contribution). This instance represents the candidate architecture shown in Figure 18.

Figure 21: Quad chart displaying all of the combinations of system functions and design patterns not included in the candidate architectures generated by the prototype decision support system for the *Blue Perspective* and the *Red Perspective*.

Those combinations not selected as part of the *Blue Perspective* are represented by blue diamonds. Those combinations not selected as part of the *Red Perspective* are represented by red circles. The size of the glyph (diamond or circle) represents that combinations costs compared to the costs of the other combinations not included in that perspective. This instance represents the candidate architecture shown in Figure 19.

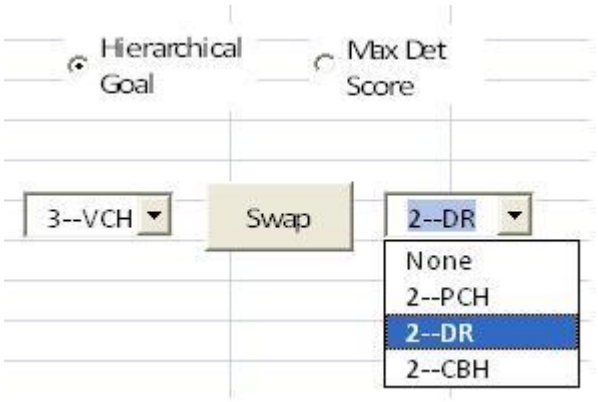


Figure 22: Set of tools the user can use to adjust the candidate architectures generated by the prototype decision support system (*Blue Perspective* and the *Red Perspective*).

This includes the ability to select an architecture to modify—Hierarchical Goal (*Blue Perspective*) or Max Det Score (*Red Perspective*)—a selected combination of system function and design pattern to remove (left of the Swap button), and a combination not included in the candidate architecture to add (right of the swap button). This instance represents the information show in Figure 18.

	Deterrence Score (Blue)	Cost (Blue)				
	13 -- 23 (-1)	9050 -- 19950 (-600)				
Higher Ranked Blue	Lower Ranked Blue	Bigger Det Red	Smaller Det Red			
4	1	2(-1)	3(+1)			
Not Included	Included	Swaped Out	Swaped In			
		Hierarchical Goal Ranking (Blue)				
ID	System Function	Design Pattern	*Rank*	*Deterrence Score*	*Cost*	*Value Factors*
3--VCH	System Control	Virtual Configuration Hopping	3	3	3000	Deterrence; Restoration
3--DR	System Control	Diverse Redundancy	3	3	2000	Restoration
3--SV	System Control	Secure Voting	3	2	2500	Deflection; Restoration
2--PCH	Sensing	Physical Configuration Hopping	2	4	3000	Deterrence; Restoration
2--DR	Sensing	Diverse Redundancy	2	3	4900	Restoration
2--SV	Sensing	Secure Voting	2	2	1150	Deflection; Restoration
2--CBH	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1--DR	User Display	Diverse Redundancy	1	4	1000	Deterrence

Figure 23: Illustrates how the prototype decision support system supports the user as they adjust the candidate architectures into new architectures.

In this instance the combination of system function and design pattern with ID 3—VCH was removed and the combination with ID 2—CBH was added. The changes to the effectiveness and costs of the architectures are show in summary statistics. Furthermore, these changes have been highlighted to indicate (potentially) positive (green) or negative (red) changes.

3.3 USE CASE

This tool was utilized to support the design process for Prototype Application of System Aware Cyber Security Surveillance Mission on an Unmanned Air Vehicle project. In addition to supporting the derivation of the architecture for this project, this activity afforded the opportunity to learn more about the benefits and limitations of the scoring framework approach.

The decision support tools were able to support a useful way of designing and selecting an architectural solution for the project. This included the ability to aid in the identification of important system functions, as well as, in the selection—and creation—of System-Aware design patterns to protect those system functions. In addition, several additional areas for improvement were suggested. First, the prototype system assumed that all system functions were prioritized only according to the possible consequences that could result from a cyber attack; however, initial usage suggested that the prioritization of a system's functions is, in fact, a combination of multiple factors. For example, during the discussion, all system functions were found to classified into three broad categories, (1) functions related to the system's operations (e.g., navigation), (2) functions used to carry out the system's specific mission objectives (e.g., radar), and (3) functions related to the system operators ability to control the system (e.g., the ability to set waypoints). Some members considered those functions related to the performance of the system as most critical: believing that these functions could be used to cause catastrophic damage to the entire system and compromise the mission. Other members disagreed with this assessment. They believed that such attacks could be easily identified and deflected by other means—such as operator intervention—and, furthermore, may degrade the system's functionality but not prevent it from accomplishing its mission. Instead, these members advocated that those system functions that were directly related to the mission and could be compromised in ways that were difficult to detect were most important to protect, as the consequences of such an attack might not result in catastrophic losses, but could result in mission failure. In addition, as these attacks would be difficult to detect, they could persist for multiple missions. For example, a cyber attack against a UAV's engine could result in the aircraft crashing—a catastrophic loss. However, the loss of an engine might be detected by the operator who could possibly take action to ensure the UAV's mission was completed before it crash landed. Alternatively, if the cyber attack compromised the UAV's radar system and resulted in it reporting misleading information, the UAV would be in no danger of loss, but its mission of detection and scouting might be compromised. In addition, as the radar is reporting realistic, and false information, such tampering may not produce any obvious signs and be difficult to detect; thus, persisting for an extended time frame. Finally, all members noted that those system functions that would be built from more stable components should be ranked higher; i.e., some system functions may be built using components that will be in service for years, while others may be built from components that will be upgraded frequently. Those system functions that will be upgraded frequently should receive a lower priority as the constant upgrading would provide a certain degree of protection by possibly deterring an adversary.

Currently the prototype system only provides the cyber attack assessment (red) team with the ability to assign an integer value of one to four to the security afforded by each system function design pattern solution. During the discussion, it was suggested that a larger range of values be available to allow the red team to more accurately assess the security offered by a given solution.

SYSTEMS ENGINEERING
Research Center
A US DoD University Affiliated Research Center

System Aware Cyber Security UAV Application Project

Barry Horowitz

November, 2012

1

What Is System Aware Cyber Security?

SYSTEMS ENGINEERING
Research Center

System Aware Cyber Security

- Operates at the system *application-layer*,
  - For *security inside* of the network and perimeter protection provided for the whole system
  - Directly protects the *most critical system functions*
  - Solutions are *embedded within* the protected functions
- Addresses *supply chain* and *insider threats*
- Includes *physical systems* as well as *information systems*
- Solution-space consists of *reusable design patterns*, reducing unnecessary duplications of design and evaluation efforts
- Includes a *scoring framework* for supporting Systems Engineers in evaluating alternative architectures

SYSTEMS ENGINEERING
Research Center

Broad Objective

Reversing cyber security asymmetry from favoring our adversaries (small investment in straightforward cyber exploits upsetting major system capabilities), to favoring the US (small investments for protecting the most critical system functions using System Aware cyber security solutions that require very complex and high cost exploits to defeat)

SYSTEMS ENGINEERING Research Center

Example System Aware Design Patterns

- **Diverse Redundancy** for post-attack restoration
- **Diverse Redundancy + Verifiable Voting** for trans-attack attack deflection
- **Physical Configuration Hopping** for moving target defense
- **Virtual Configuration Hopping** for moving target defense
- **Data Consistency Checking** for data integrity and operator display protection
- **Physical Confirmations of Digital Data** for data integrity
- **Use of Analog Components** for diversely redundant solutions

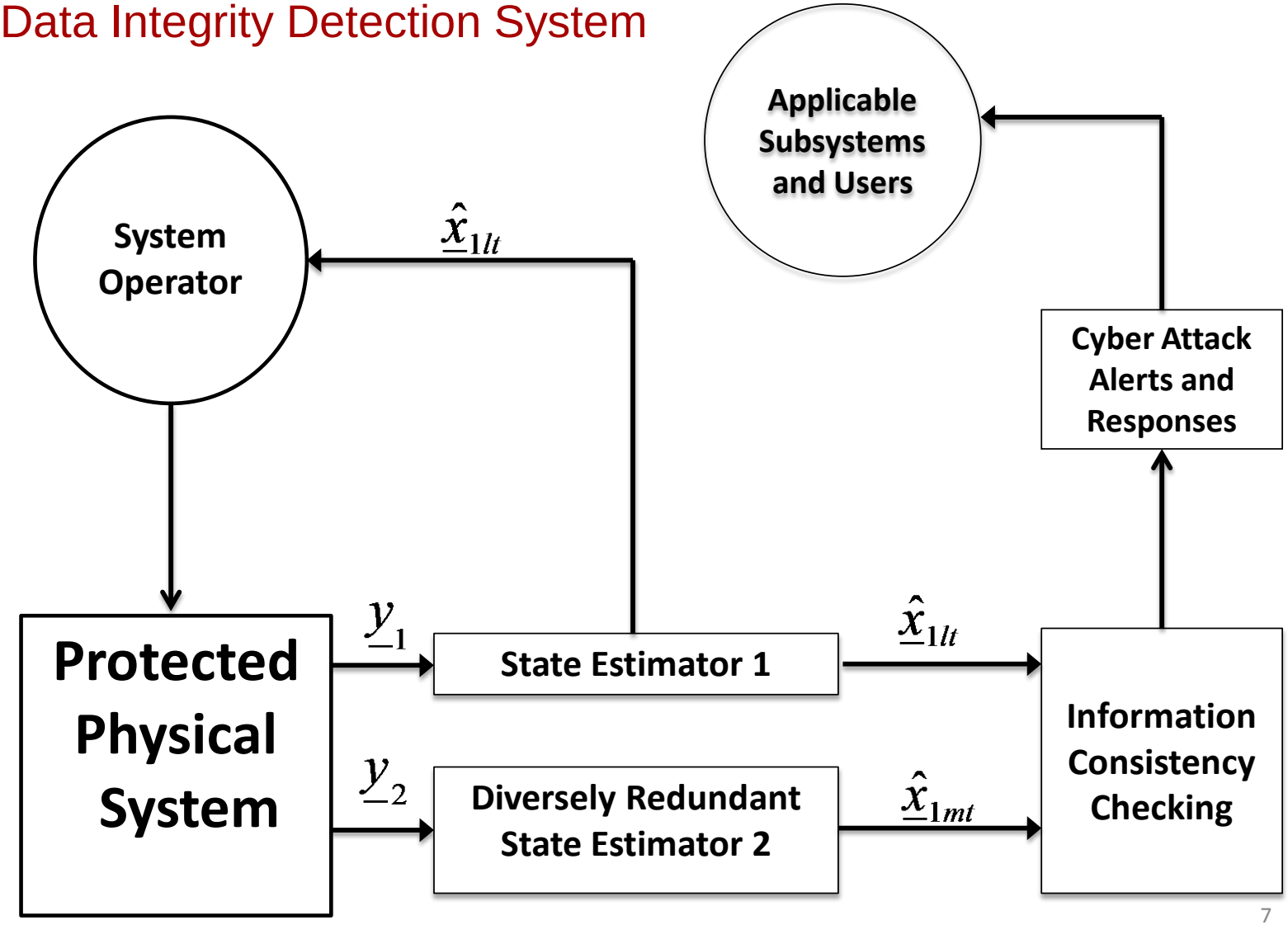
EXAMPLE STUXNET-LIKE APPLICATION

Design Patterns

Diverse Redundancy

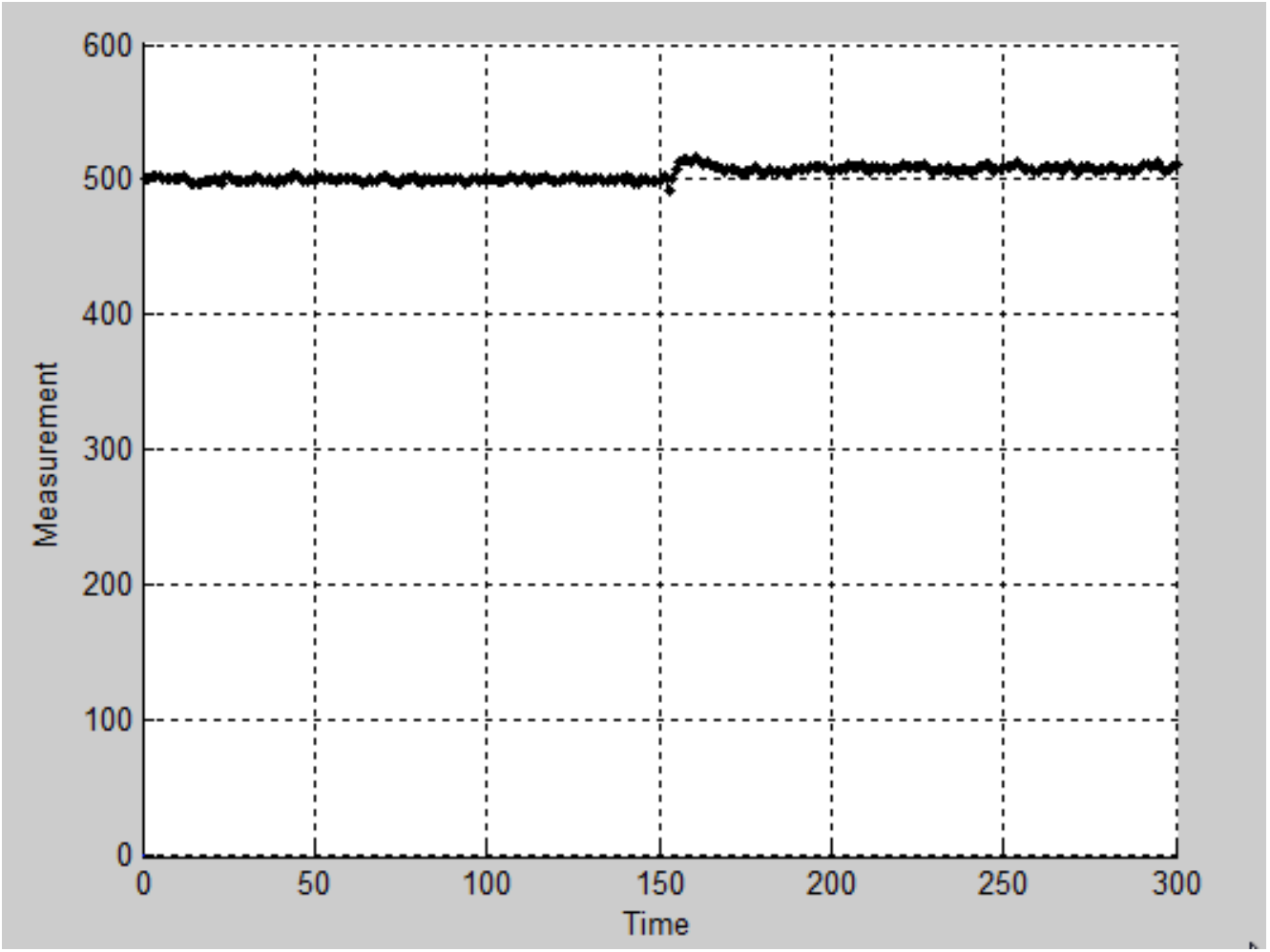
Data Consistency Checking

Simplified Block Diagram for Inference-Based Data Integrity Detection System



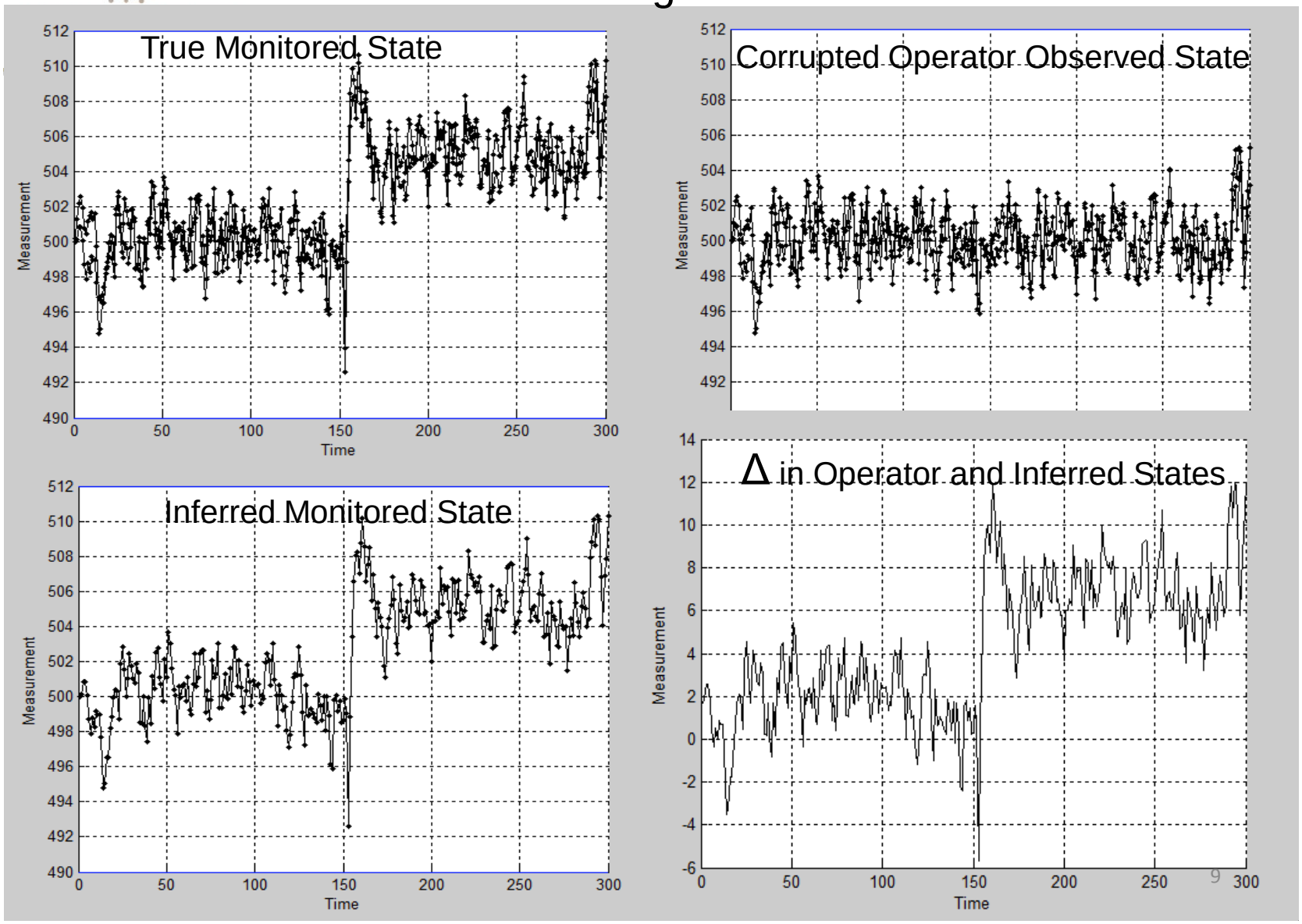
SYSTEMS ENGINEERING
Research Center

Simulated System Output Based Upon Controller Attack



8

Simulated Regulator Attack



SYSTEMS ENGINEERING Research Center

System Aware Security Architecture Design Process

- Identify and prioritize critical system functions to protect — Blue Team
- Identify candidate highly asymmetric attack vectors — Red Team
- Select multiple design patterns for each protected function — Blue Team
- Determine architectures within specific defender budgets — Green Team
- Select specific architecture based on comparison of evaluations of the defenders' cost to protect versus change in attackers' costs to develop and evaluate new exploits – (Blue/Red/Green Teams)

UAV Project

SYSTEMS ENGINEERING
Research Center

Our Team

- UVa
  - Developed the System Aware cyber sec concept
  - Developed an initial set of reusable design patterns
  - Performed system analyses related to design pattern performance
  - Developed the cyber security scoring framework
- GTRI
  - UAV system integration
  - Sensors and sensor signal processing
  - System testing and evaluation

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Outlaw Performance Characteristics

Parameter	Value
Max Takeoff Gross Wt.	150 lbs
Wing span	16 ft
Payload capacity	35 lbs
Endurance	1-2 hrs
Cruise speed	75 kts
Installed power	16.5 hp

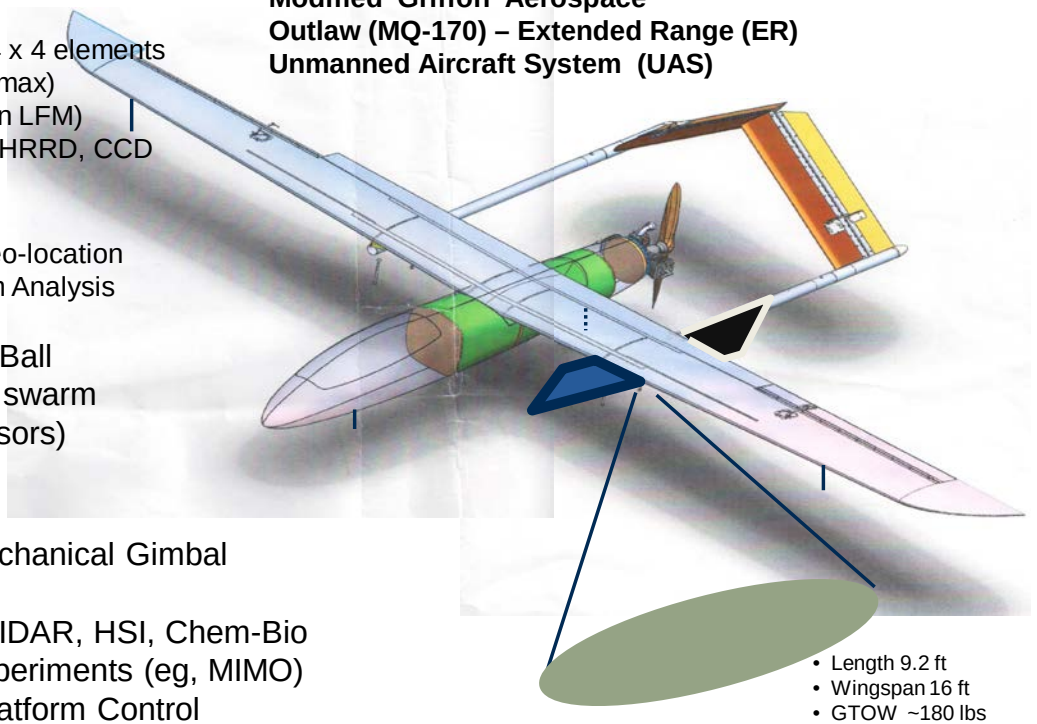
GAUSS

GTRI AIRBORNE UNMANNED SENSOR SYSTEM

FOUR SENSOR OBJECTIVE BASELINE

- Multi-Channel Radar (8 channels)
  - ESA Antenna: 8 phase centers, each 4 x 4 elements
  - X-band, 600 MHz BW (design; 1 GHz max)
  - Arbitrary Waveform Capable (1st design LFM)
  - Acquisition Modes: DMTI, SAR, HRR, HRRD, CCD
- Multi-Channel SIGINT
  - Near 1 and 2 GHz Bands
  - Two orthogonal dipole pairs: TDOA geo-location
  - Ambient Complex-Baseband Spectrum Analysis
  - Signal Copy Selected Sub-Bands
- Gimbaled, Stabilized EO/IR Camera Ball
- High Precision GPS & INS (eventual swarm capable inter-UAV coherent RF sensors)

Modified Griffon Aerospace
Outlaw (MQ-170) – Extended Range (ER)
Unmanned Aircraft System (UAS)

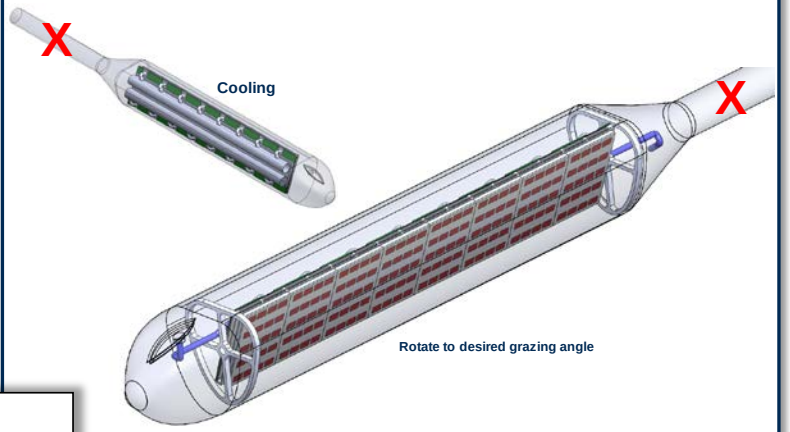


- Length 9.2 ft
- Wingspan 16 ft
- GTOW ~180 lbs
- Payload ~35-40 lbs
- Ceiling 14 kft
- Cruise speed 70 knts
- Endurance 9 hrs

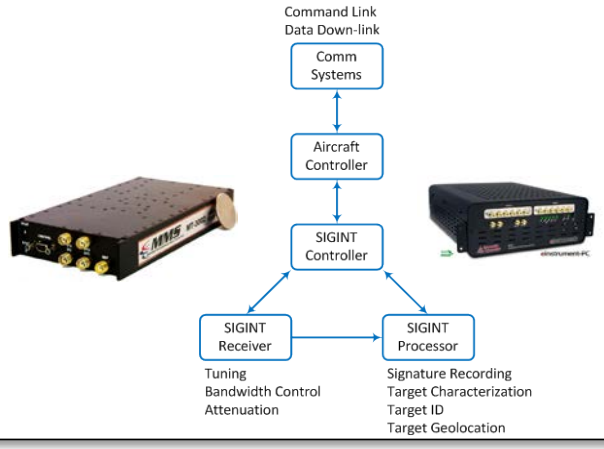
CAPABILITIES

- Electronic Scanning; No Antenna Mechanical Gimbal
- Multi-TB On-Board Data Recording
- Reconfigurable for Other Sensors: LIDAR, HSI, Chem-Bio
- Multi-Platform Distributed Sensor Experiments (eg, MIMO)
- Autonomous & Collaborative Multi-Platform Control
- Space for Future GPU/FPGA On-Board Processing

RADAR POD ANTENNA ASSEMBLY



SIGINT System



CLOUD CAP EO CAMERA

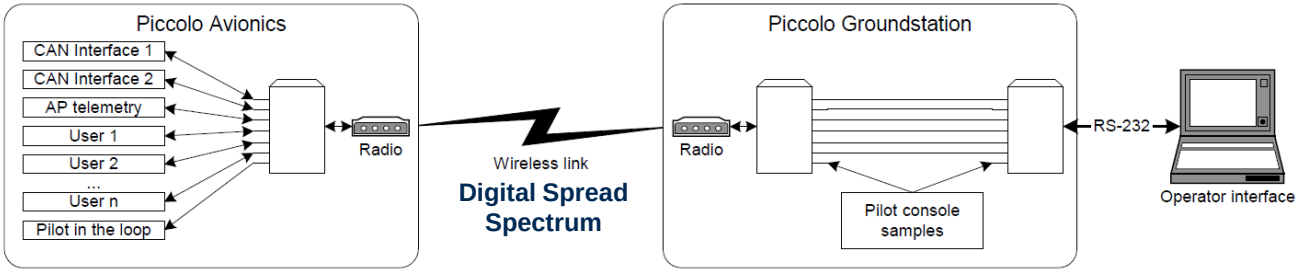
- Stabilized camera micro-gimbals
- 2- axis stabilization; optional elec. stab.
- Object and scene tracking
- LWIR and SWIR cameras available
- Laser pointer and ranging available
- 0.05 deg pointing resolution
- 10W typ, 18W max
- 2 lbs
- ViewPoint software can geo rectify to map and perform image mosaicing
- Links through Piccolo autopilot
- Griffon has retractable configuration

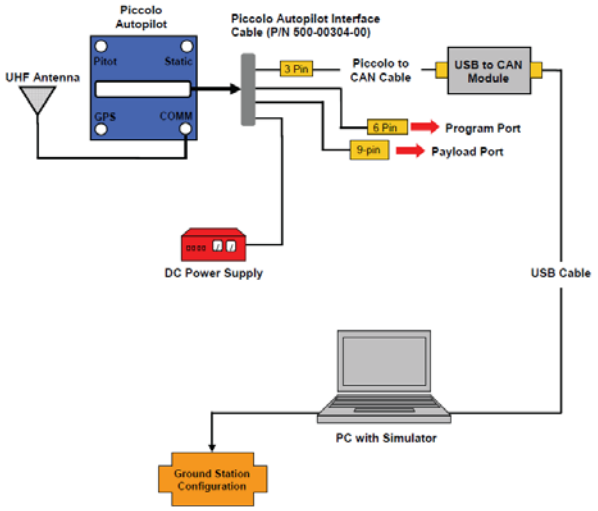
TASE 150

15

Piccolo Auto-Pilot

Piccolo communication architecture





Critical System Functions to Protect against Highly Asymmetric Attacks

SYSTEMS ENGINEERING Research Center

Initial System Aware Architectural Assessment

- Three categories of critical system functions to protect:
  - Platform Subsystems (platform control, navigation, mission control, air/ground comm.),
  - Sensor subsystems,
  - Human support subsystems
- Most highly asymmetric attack risks – **System parameter changes** (e.g., waypoint changes, flight control system changes, surveillance mode changes, signal processing changes), GPS navigation system corruption
- Which design patterns – Control parameter assurance (flight stability control, field of view control, etc), waypoint assurance, navigation system assurance using existing diverse navigation sources, sensor pointing assurance, security control channel using analog spread spectrum radio communication for air/ground communications
- Types of Evaluations – Simulation, Rapid prototyping (version 1), HW/SW in the loop emulation evaluations, Rapid Prototyping (version 2) with live flight. Requires metric development and corresponding measurement capabilities- ground and air
- Architecture Decision support – Blue, Red, Green team symmetry analyses

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Exploitable Parameters and Commands (1)

Group	Parameter	Purpose	Possible exploitation
Actions	Abort	action depends on current aircraft state	
	Engine On	allows user to enable or disable engine	could be used to disable the engine
Flight plan	waypoint coordinates	control flight path of aircraft	altitude can be zeroed, waypoint can be moved
Command limits	dynamic pressure limit	limit aircraft airspeed	Setting limit to low value could be used to stall aircraft
	pressure altitude limit	limit aircraft altitude	Setting limit to low value could be used to ground aircraft
	bank angle limit	limit aircraft turn rate	extending limit beyond 30 deg could cause aircraft to stall

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Exploitable Parameters and Commands (2)

Group	Parameter	Purpose	Possible exploitation
Output limits	aileron travel limits	limits autopilot cmds to ailerons	setting limits to zero could shut down cmds to actuator
	elevator travel limits	limits autopilot cmds to elevator	setting limits to zero could shut down cmds to actuator
	rudder travel limits	limits autopilot cmds to rudder	setting limits to zero could shut down cmds to actuator
	throttle travel limits	limits autopilot cmds to throttle	setting limits to zero could shut down cmds to actuator
	flap travel limits	limits autopilot cmds to flaps	setting limits to zero could shut down cmds to actuator

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Exploitable Parameters and Commands (3)

Group	Parameter	Purpose	Possible exploitation
Mission limits	pilot timeout	controls when system switches to autopilot when there are no manual inputs	setting limit to zero will block possibility of manual flight
	comm timeout	controls when Lost Comm Waypoint plan is executed after comms are lost	setting limit to low value could cause mission aborts
	GPS timeout	defines the amount of time for the aircraft to continue to perform normally without a new GPS solution	setting limit to low value could cause mission aborts

Example Parameter Control-Based Threat Implementation

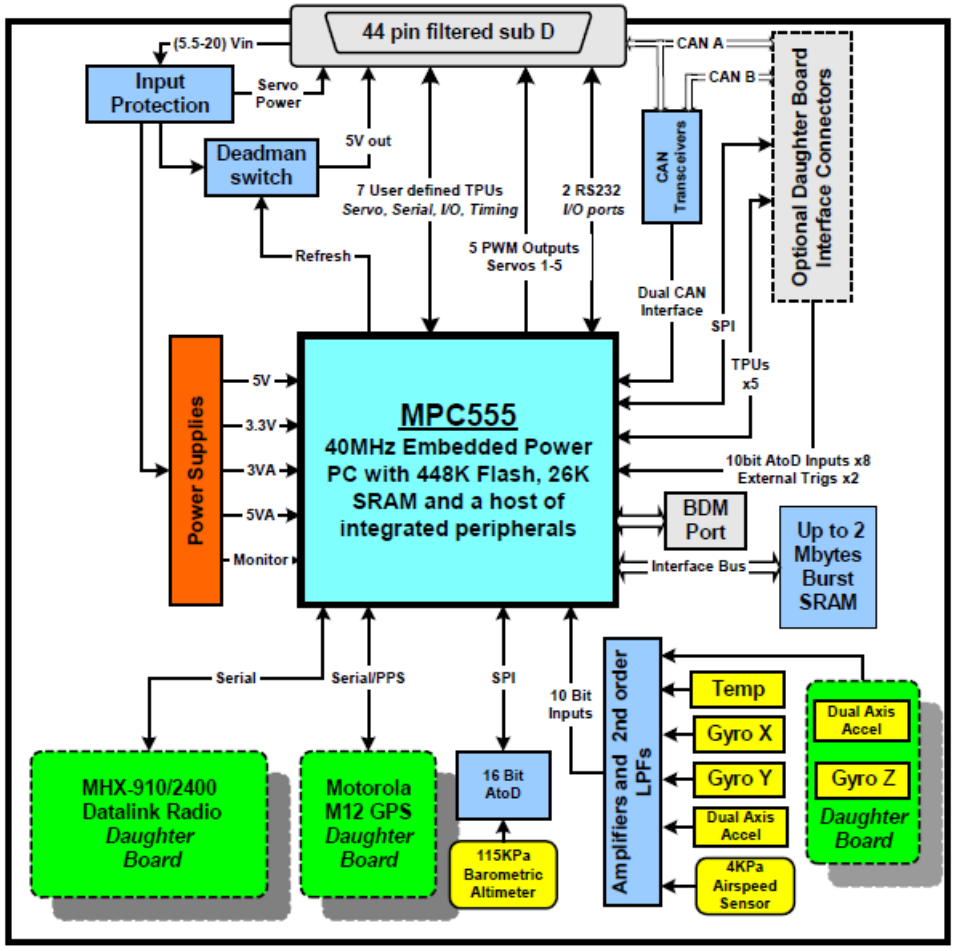
SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Piccolo-Based On-Aircraft Exploit for Parameter Change Attack

- Parameter modification attack exploit
  - During pre-flight, a user password is set within Piccolo system to lock key flight parameters
    - Exploit detects and stores this password
  - Whenever aircraft enters a designated region, the parameter adjustment part of the exploit will be triggered – Based on reading GPS measurements onboard the aircraft
  - Once initiated, the exploit will
    1. Unlock key flight configuration parameters
    2. Modify the attacker selected subset of the flight configuration parameters
    3. Prevent subsequent ground operator changes, by locking the key flight configuration parameters with a newly generated password
  - Once outside the designated region, the exploit will reset password and parameters to mask the attack having occurred

SYSTEMS ENGINEERING
Research Center
A US DoD University Affiliated Research Center

Autopilot Block Diagram

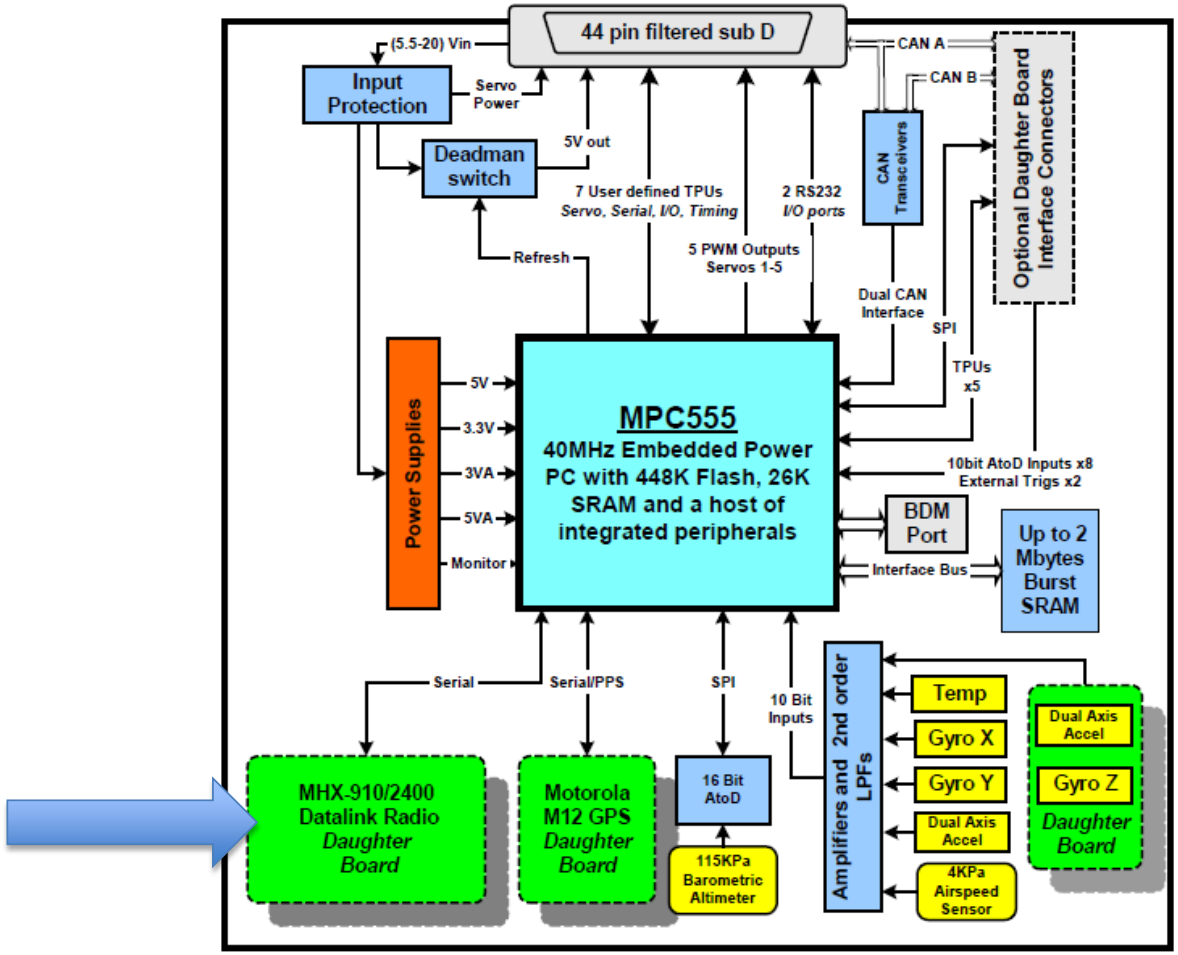


SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Exploited Autopilot Block Diagram

Data Link Board Exploit

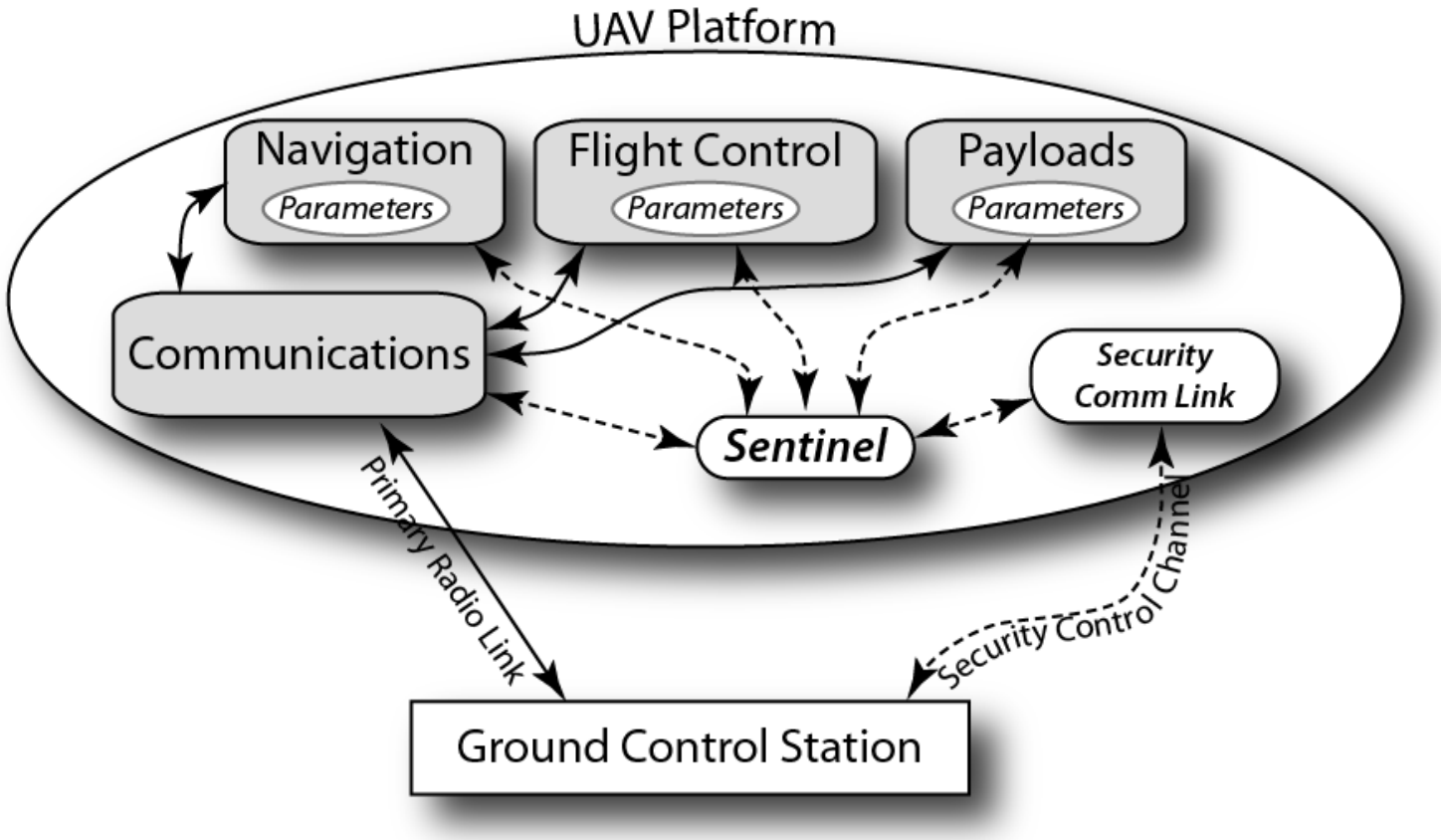
- Read Password on Uplink
- Read GPS on Down Link
- Modify Parameters
- Change Password



System Aware Implementation Framework

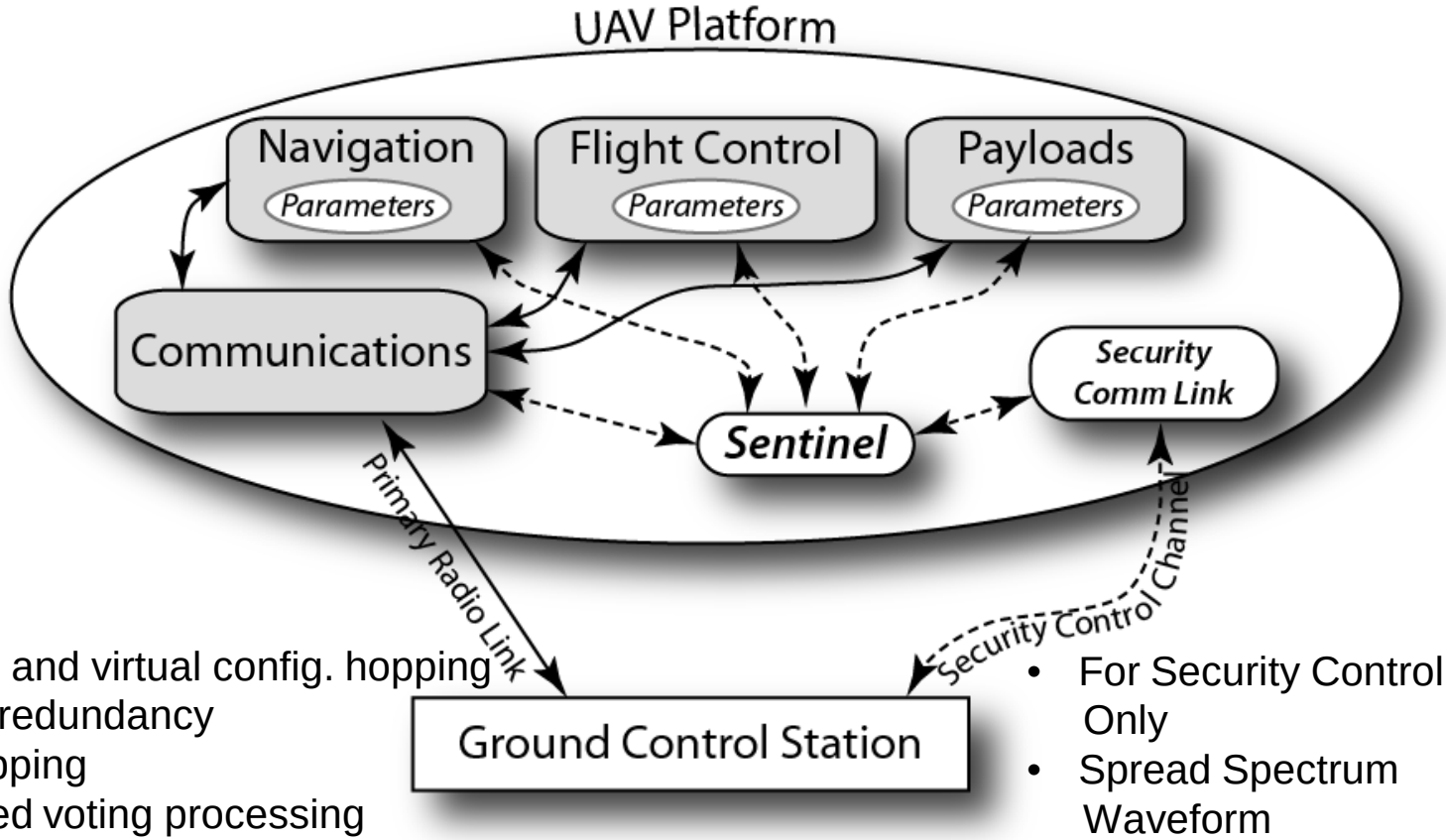
SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

UAV Platform Augmentations



SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

UAV Platform Augmentations



Physical and virtual config. hopping
Diverse redundancy
Port Hopping
Dedicated voting processing
Software power utilization-based verification
CPU and memory usage-based verifications

- For Security Control Only
- Spread Spectrum Waveform
- Low Data Rate

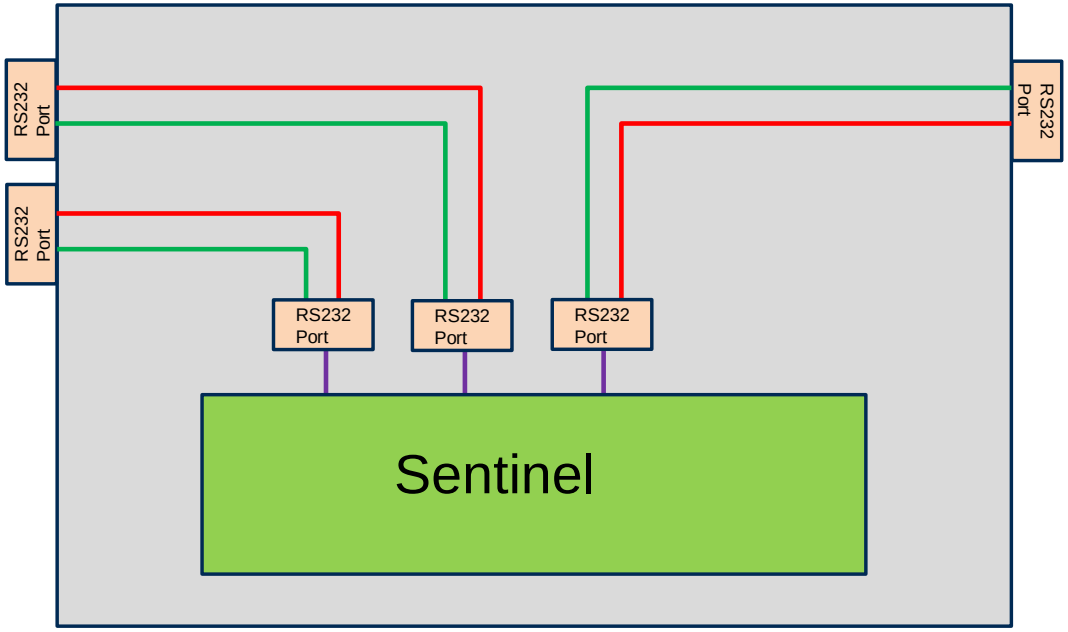
SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

On-Aircraft, Sentinel-Based Solution

- Detection
  - Sentinel, at regular intervals, will monitor stability of parameters
  - Coordination with the ground system will occur through the low-data rate secure radio connection
- Restoration
  - Sentinel resets parameters through the MPC55 processor
  - The Sentinel also resets the Parameter Control password
    - Sends required information back to ground through secure channel
    - Locks out exploit in the process.

SYSTEMS ENGINEERING
Research Center
A US DoD University Affiliated Research Center

RS232 -Based Sentinel



- Actively monitors multiple RS232 data sources (GPS Information, Password, Parameter Settings)
- Provides voting and intervention capability

SYSTEMS ENGINEERING
Research Center

Additional Threats and Solutions

- GPS navigation system attack – Piccolo includes INS and barometric altimeter as diversely redundant sources for data consistency checking; Cameras include INS for pointing purposes, providing another diversely redundant measurement source.
- Attack on antenna beam pointing – Use of Doppler effects on signal returns and aircraft navigation information to confirm relationship between antenna beam pointing direction and aircraft velocity vector
- Attack on Ground site displays causing important operator error – Use of Security Data Link in the Sentinal to confirm air and ground-based information prior to operators taking mission critical actions

SYSTEMS ENGINEERING
Research Center
A US DoD University Affiliated Research Center

Schedule

- Two Year effort to achieve a flight tested implementation of a highly secured Sentinel providing mission critical cyber security, including a ground-based emulation support capability for pre-flight test use.
- Includes demonstration of exploits successfully utilized and protection against them
- Development and documentation of new design patterns, as required
- Advancement of architectural design approach with corresponding improvements in decision support tools

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Future Work

- Pursuing other areas of application for advancing the development and use of System Aware security
- Future areas of importance include:
  - Broader set of unmanned vehicles
  - Wider set of mission applications
  - Big Data applications for data integrity and model integrity checking
  - Support to management of offensive cyber attack selections, by predicting possible adversary responses to US attacks

SYSTEMS ENGINEERING
Research Center
A US DoD University Affiliated Research Center

System Aware Cyber Security UAV Application Project UVA/GTRI

Project Leader: Barry Horowitz, UVA
Co-Project Leader: Bill Melvin, GTRI

January, 2013

1

SYSTEMS ENGINEERING
Research Center

Broad Objective

Reversing cyber security asymmetry from favoring our adversaries (small investment in straightforward cyber exploits upsetting major system capabilities), to favoring the US (small investments for protecting the most critical system functions using System Aware cyber security solutions that require very complex and high cost exploits to defeat)

SYSTEMS ENGINEERING
Research Center

System Aware Cyber Security

- Operates at the system *application-layer*,
  - For *security inside* of the network and perimeter protection provided for the whole system
  - Directly protects the *most critical system functions*
  - Solutions are *embedded within* the protected functions
- Addresses *supply chain* and *insider threats*
- Includes *physical systems* as well as *information systems*
- Solution-space consists of *reusable design patterns*, reducing unnecessary duplications of design and evaluation efforts
- Design Patterns can be implemented in a super secure programmable Sentinel (S3)
- Geographically separated system locations (e.g., UAV air and ground sites) can include coordinating Sentinels over a dedicated secure communications link
- Includes a *scoring framework* for supporting Systems Engineers in evaluating alternative architectures

SYSTEMS ENGINEERING Research Center

Integration of Fault Tolerance, Automatic Control and Information Assurance

- What's Different for each technology community
  - Fault Tolerance
    - Asymmetric attacks vs random failures
    - Synchronized dependent attacks on system components vs random coupling of independent failures
    - Time varying, situation-related, attacks vs random intermittent failures
    - False alarms emerging as a potential National Security issue in Critical Infrastructure Systems
  - Automatic Control
    - High rates of system reconfiguration
    - Acceptable state estimation performance for detecting and responding to cyber attacks
    - Semi-automatic solutions
  - Information Assurance
    - System Aware
    - Collateral, system-specific, performance impacts of embedded security solutions
- Plus:
  - Require secure implementation of solutions

SYSTEMS ENGINEERING Research Center

Example System Aware Design Patterns

- **Diverse Redundancy** for post-attack restoration
- **Diverse Redundancy + Verifiable Voting** for trans-attack attack deflection
- **Physical Configuration Hopping** for moving target defense
- **Virtual Configuration Hopping** for moving target defense
- **Data Consistency Checking** for data integrity and operator display protection
- **Physical Confirmations of Digital Data** for data integrity

SYSTEMS ENGINEERING Research Center

System Aware Security Architecture Design Process

- Identify and prioritize critical system functions to protect — Blue Team
- Identify candidate highly asymmetric attack vectors — Red Team
- Select multiple design patterns for each protected function — Blue Team
- Determine architectures within specific defender budgets — Green Team
- Select specific architecture based on comparison of evaluations of the defenders' cost to protect versus change in attackers' costs to develop and evaluate new exploits – (Blue/Red/Green Teams)

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Overall Proposal Through FY 14

- Two year effort to achieve a live-flight evaluated implementation of a highly secured Sentinel providing mission critical cyber security, including ground-based emulation support capabilities for pre-flight test use
- Includes demonstration of successful exploits and protection against them
- Development and documentation of new design patterns, as required, including human-in the-loop patterns
- Advancement of an architectural design approach that includes geographically distributed Sentinels with corresponding improvements in decision support tools
- Reusable open architecture emulation capability for evaluating new design patterns in different application settings

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Prototype Development and Evaluation Plan

- Two prototypes: Ground (FY 13-14) & Air (FY14) – Option to start Air efforts in FY13
- Ground serves as an initial version for integration with both Piccolo and Mission Equipment
  - Serves as a planning and evaluation vehicle for developing a functional Sentinel prototype (exploits, multiple design patterns)
  - Includes introduction of 2 coordinating Sentinels (air and ground) for the UAV application
- Air includes:
  - Onboard implementation of attack exploits
  - Onboard design pattern implementation, employing separate partial implementations due to SWAP constraints

Initial System Prototype

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Outlaw Performance Characteristics

Parameter	Value
Max Takeoff Gross Wt.	150 lbs
Wing span	16 ft
Payload capacity	35 lbs
Endurance	1-2 hrs
Cruise speed	75 kts
Installed power	16.5 hp

Selection of Critical System Functions to Protect against Highly Asymmetric Attacks

SYSTEMS ENGINEERING Research Center

Initial System Aware Architectural Assessment

- Considered three (3) categories of critical system functions to protect:
  - Platform Subsystems (platform control, navigation, mission control, air/ground comm.)
  - Sensor subsystems
  - Human support subsystems
- All of the considered system categories have general applicability beyond the UAV application, regarding the potential nature of attacks and the corresponding risks resulting from attacks

SYSTEMS ENGINEERING
Research Center

Initial System Aware Architectural Assessment

- Most highly asymmetric attack risks – **System parameter changes** (e.g., waypoint changes, flight control system changes,, signal processing changes), GPS navigation system corruption, surveillance camera pointing control
- Which design patterns – Control parameter assurance (flight stability control, field of view control, etc), waypoint assurance, navigation system assurance using existing diverse navigation sources, sensor pointing assurance, security control channel for air/ground communications, human-in-the loop
- Types of Evaluations
  - Simulation,
  - Rapid prototyping (version 1), HW/SW in the loop emulation evaluations,
  - Rapid Prototyping (version 2) with live flight.
  - Requires metrics and corresponding measurement capabilities
- Architecture Decision support – Blue, Red, Green team symmetry analyses

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Exploitable Piccolo Parameters and Commands (1)

Group	Parameter	Purpose	Possible exploitation
Actions	Abort	action depends on current aircraft state	
	Engine On	allows user to enable or disable engine	could be used to disable the engine
Flight plan	waypoint coordinates	control flight path of aircraft	altitude can be zeroed, waypoint can be moved
Command limits	dynamic pressure limit	limit aircraft airspeed	Setting limit to low value could be used to stall aircraft
	pressure altitude limit	limit aircraft altitude	Setting limit to low value could be used to ground aircraft
	bank angle limit	limit aircraft turn rate	extending limit beyond 30 deg could cause aircraft to stall

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Exploitable Piccolo Parameters and Commands (1)

Group	Parameter	Purpose	Possible exploitation
Actions	Abort	Action depends on current aircraft state	
	Engine On	Allows user to enable or disable engine	Engine could be disabled
Flight plan	Waypoint Coordinates	Controls flight path of aircraft	Altitude can be zeroed, waypoint can be moved
Command Limits	Dynamic Pressure Limit	Limits aircraft airspeed	Aircraft can be stalled by setting the limit to a low value
	Pressure Altitude Limit	Limits aircraft altitude	Aircraft can be grounded by setting the limit to low value
	Bank Angle Limit	Limits aircraft turn rate	Aircraft could be stalled by setting the limit beyond 30 degrees

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Exploitable Piccolo Parameters and Commands (2)

Group	Parameter	Purpose	Possible exploitation
Output limits	Aileron Travel Limits	Limits autopilot cmds to ailerons	Setting limits to zero could shut down cmds to actuator
	Elevator Travel Limits	Limits autopilot cmds to elevator	Setting limits to zero could shut down cmds to actuator
	Rudder Travel Limits	Limits autopilot cmds to rudder	Setting limits to zero could shut down cmds to actuator
	Throttle Travel Limits	Limits autopilot cmds to throttle	Setting limits to zero could shut down cmds to actuator
	Flap Travel Limits	Limits autopilot cmds to flaps	Setting limits to zero could shut down cmds to actuator

Example Parameter Control-Based Threat Implementation

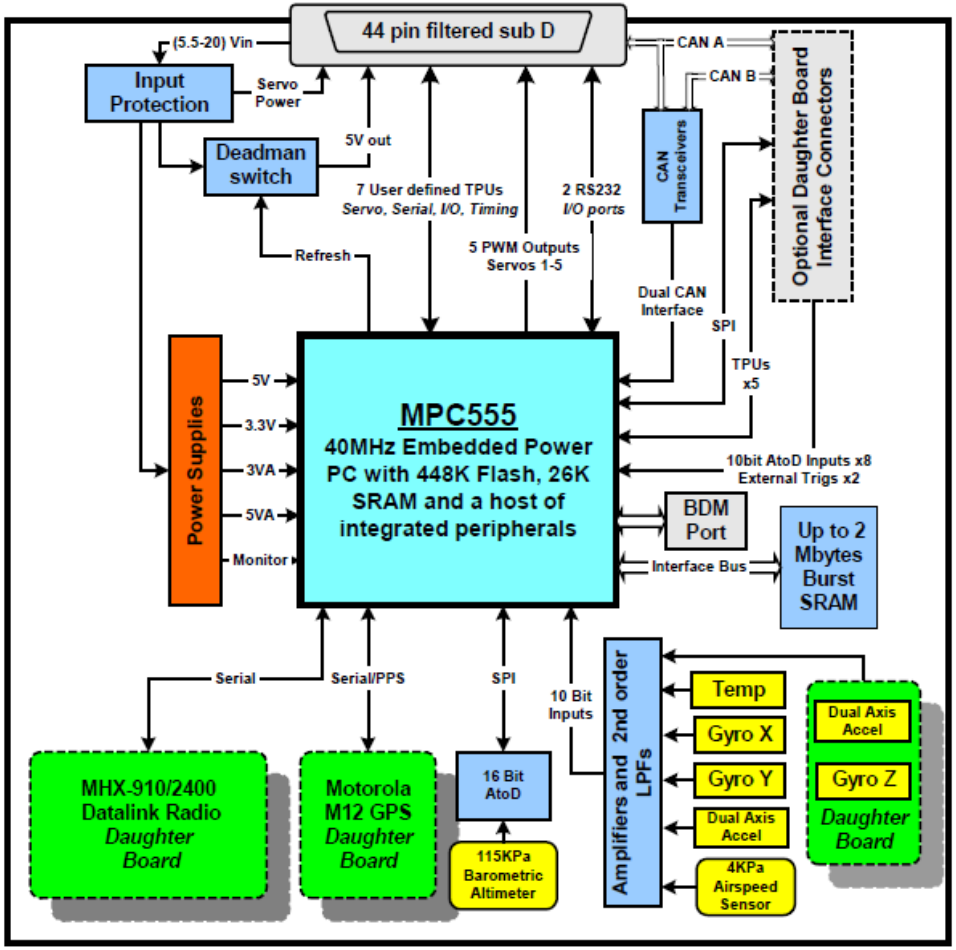
SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Piccolo-Based On-Aircraft Exploit for Parameter Change Attack

- Parameter modification attack exploit
  - During pre-flight, a user password is set within Piccolo system to lock key flight parameters
    - Exploit detects and stores this password
  - Whenever aircraft enters a designated region, the parameter adjustment part of the exploit will be triggered – based upon reading GPS measurements onboard the aircraft
  - Once initiated, the exploit will
    1. Unlock key flight configuration parameters
    2. Modify the attacker selected subset of the flight configuration parameters
    3. Prevent subsequent ground operator changes, by locking the key flight configuration parameters with a newly generated password
  - Once outside the designated region, the exploit will reset password and parameters to mask the attack having occurred

SYSTEMS ENGINEERING
Research Center
A US DoD University Affiliated Research Center

Autopilot Block Diagram

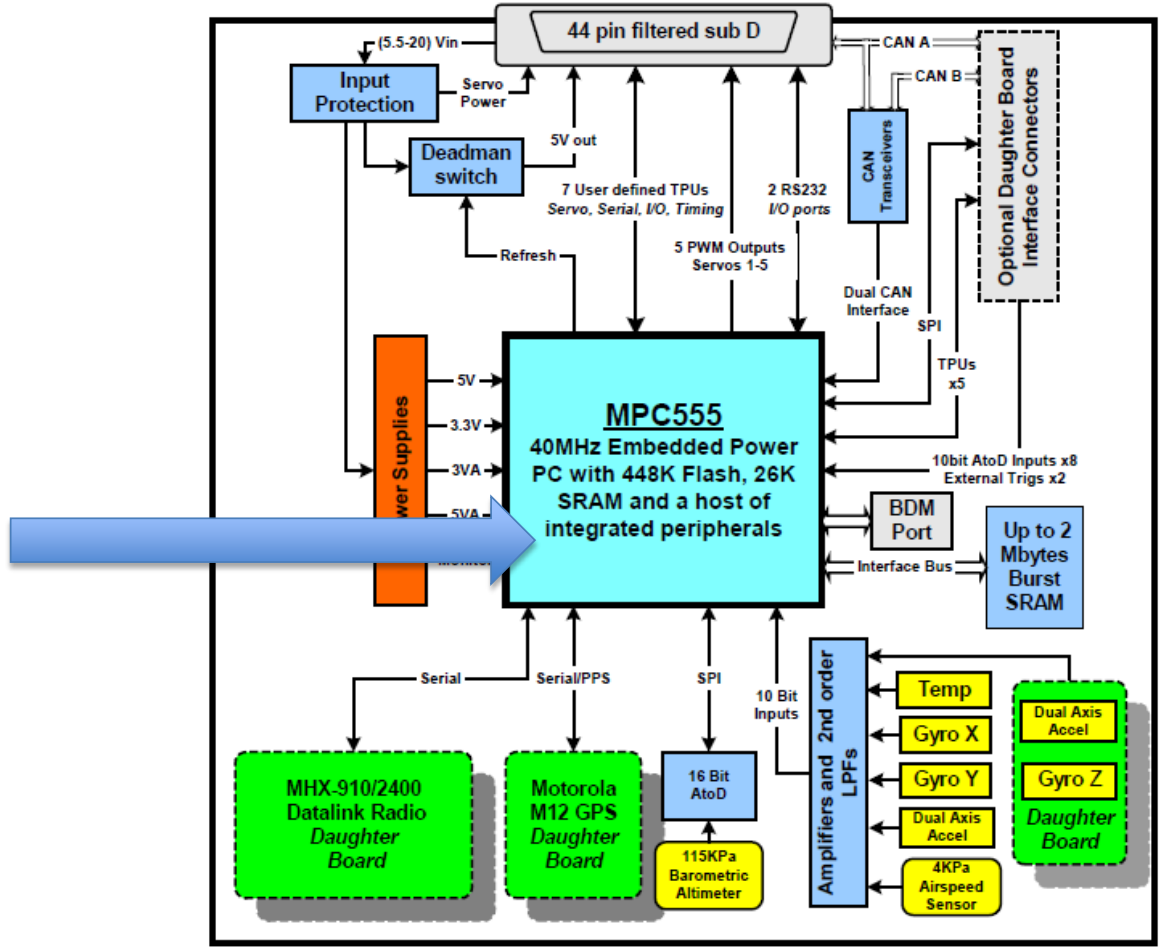


SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Exploited Autopilot Block Diagram

Processing Board Exploit

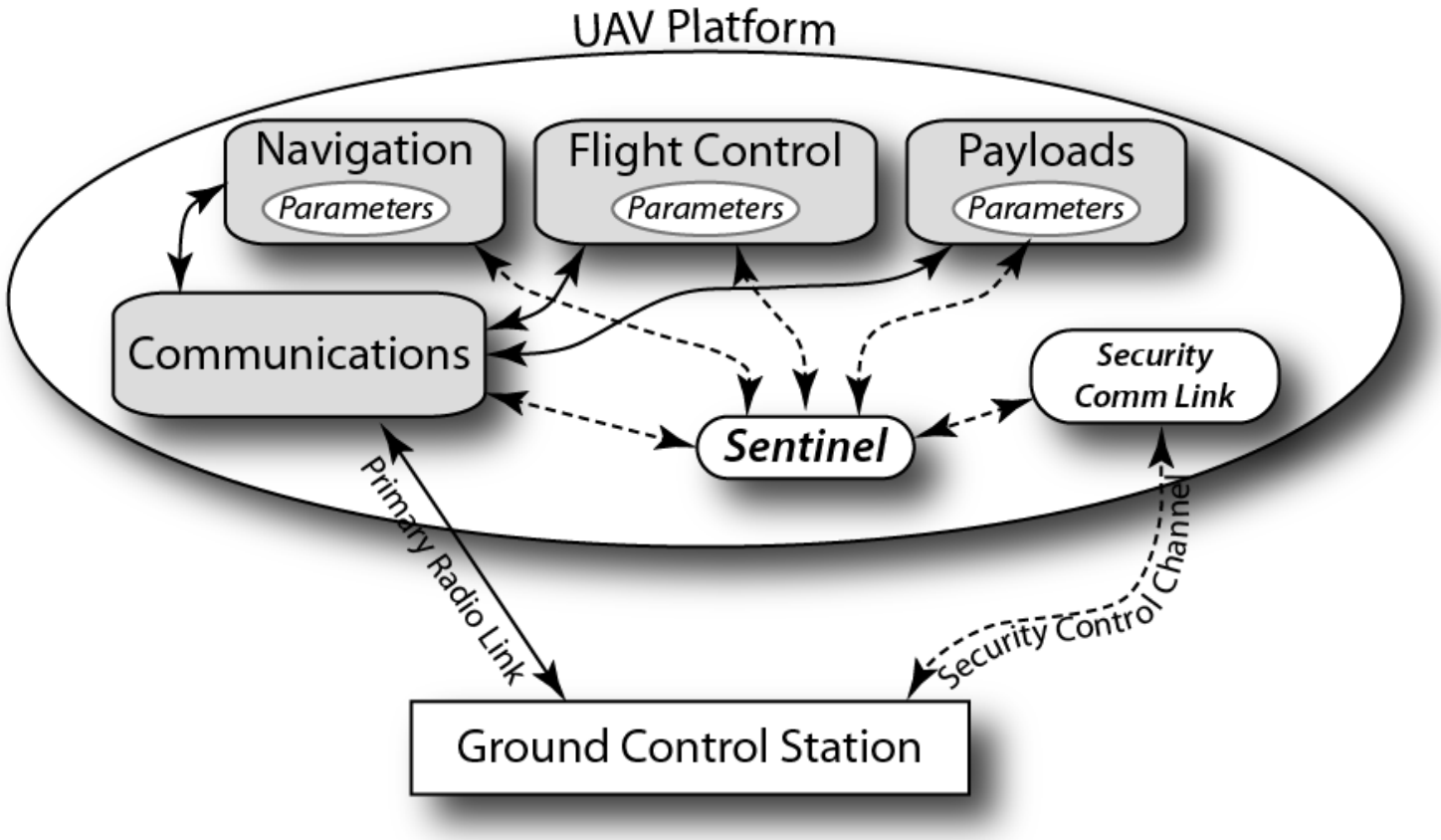
- Read Password from Uplink
- Read GPS for Down Link
- Modify Parameters
- Change Password



System Aware Implementation Framework

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

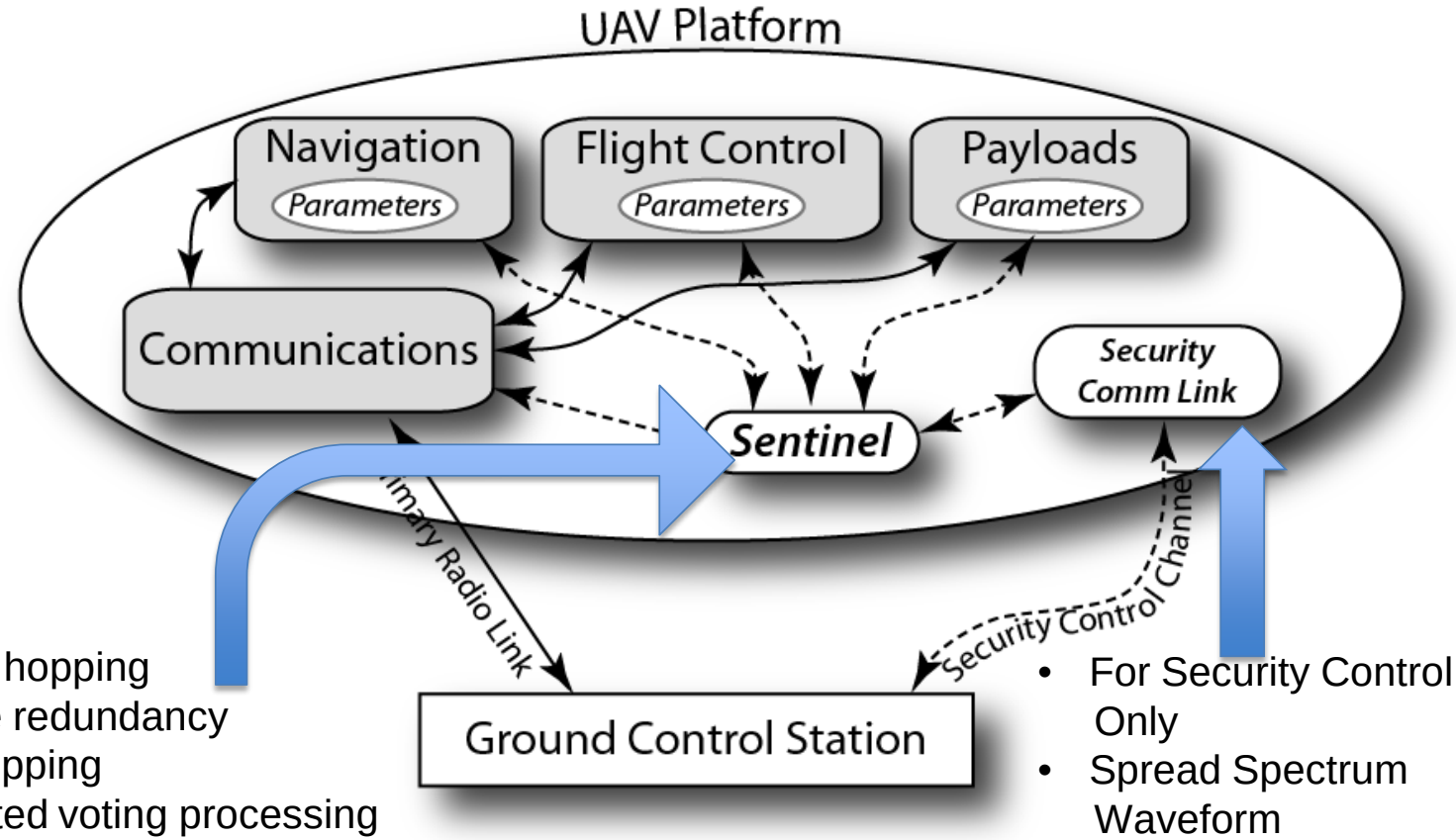
UAV Platform Augmentations



22

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

UAV Platform Augmentations



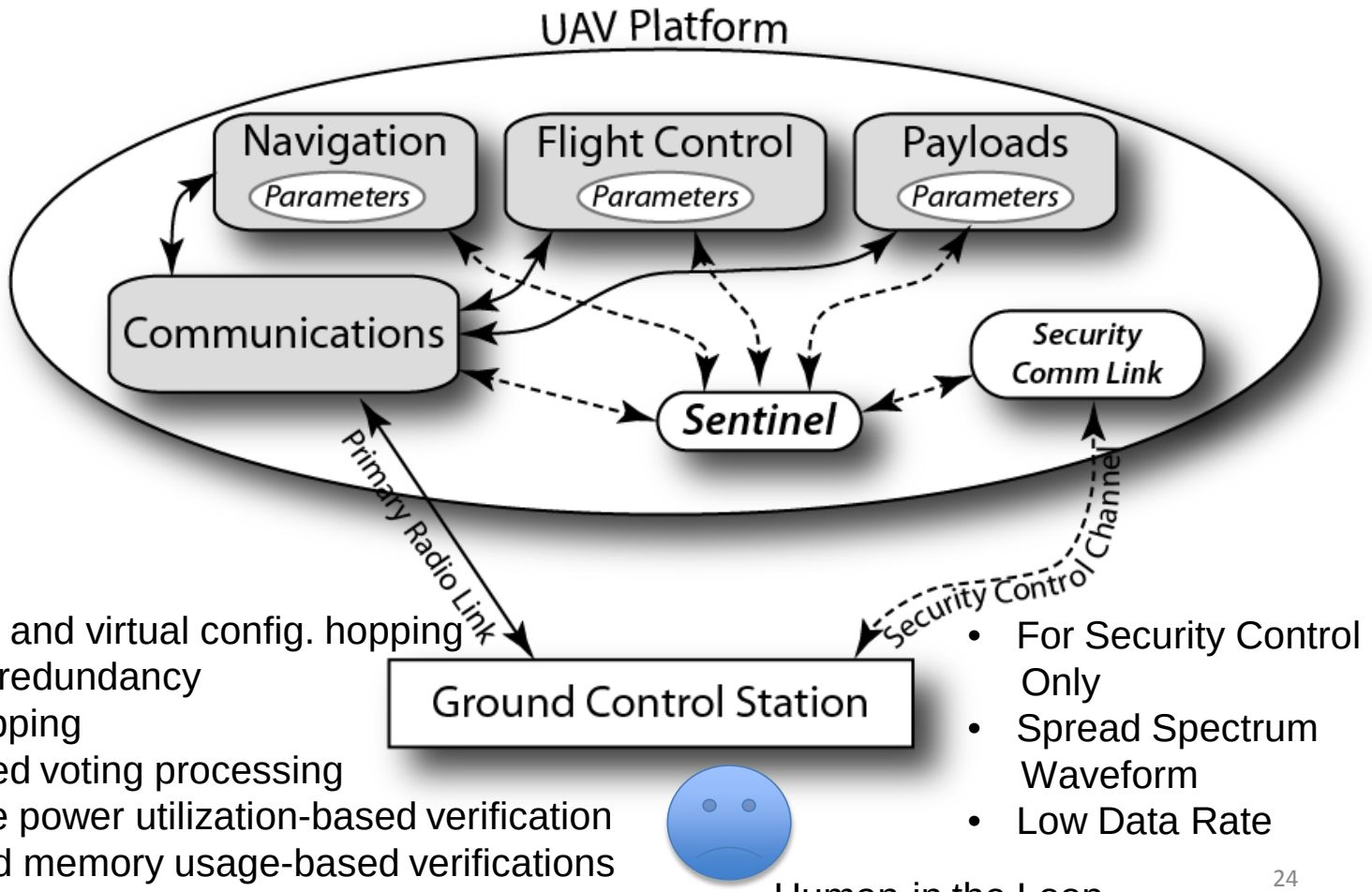
Config. hopping
Diverse redundancy
Port Hopping
Dedicated voting processing
SW power utilization fingerprint
SW CPU and memory usage fingerprint

- For Security Control Only
- Spread Spectrum Waveform
- Low Data Rate

23

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

UAV Platform Augmentations



Physical and virtual config. hopping
Diverse redundancy
Port Hopping
Dedicated voting processing
Software power utilization-based verification
CPU and memory usage-based verifications

- For Security Control Only
- Spread Spectrum Waveform
- Low Data Rate

Human-in the Loop

24

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

On-Aircraft, Sentinel-Based Solution

- Detection
  - Sentinel, at regular intervals, will monitor stability of parameters, IMU, GPS and barometric altimeter readings
  - Coordination with the ground system will occur through the low-data rate secure radio connection
- Restoration
  - Sentinel resets parameters through the MPC555 processor
  - The Sentinel also resets the Parameter Control password
    - Sends required information back to ground through secure channel
    - Locks out exploit in the process.
  - Sentinel restores navigation

SYSTEMS ENGINEERING
Research Center

Human-in-the-Loop Concept

- The Sentinel is designed to minimize false alarms due to the criticality of the system functions being protected
- Accordingly, Sentinel attack detection algorithms utilize very conservative detection thresholds that contain false alarm rates.
- These Sentinel detection thresholds are based on a statistical view, assuming the set of all mission scenarios and events
- An operator can provide the Sentinel with a queue when he or she believes that an attack may be in progress, accounting for observations unavailable to the Sentinel, and for flight mission
- The Sentinel can, in-turn, derive for the operator the likelihood of the a priori likelihood that he or she must believe is true for that attack to be detected, using the Sentinel's data
- This provides the basis for developing a human/machine interchange about the attack and its likelihood.

SYSTEMS ENGINEERING Research Center

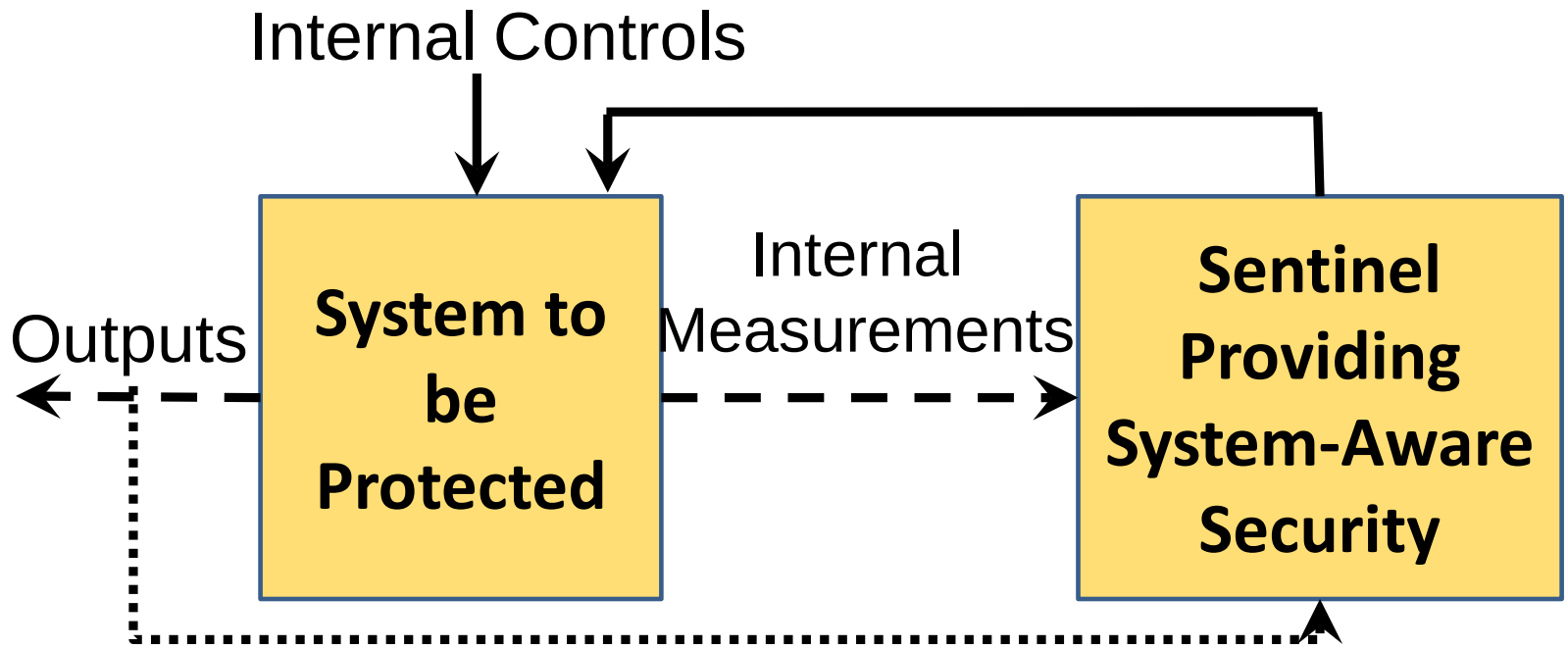
Human-in-the-Loop Dialog Concept

- Example:
  - Human – I think that my system data is being fooled with
  - Sentinel – According to my assessment there is only a 2 % likelihood of an attack in progress; too low for me to raise an alarm and switch into back-up/reduced performance mode.
  - Sentinel – If I go along with your judgment and automatically reconfigure for restoration, on a mathematical basis we'd equivalently be assuming that you believe that there is a 50% likelihood of attack, and we'd be maintaining our standard 1% false alarm rate. Do you believe that there is a 50% likelihood of an attack in progress?
  - Human – I'm not sure if there is a 50% chance, but I'd say that its somewhere between 25-50%
  - Sentinel – Let's keep monitoring for another 6 minutes and I'll have new numbers for you
  - Human – Given our mission, I can only wait 3 minutes.
  - Sentinel- OK. We'll decide what to do in 3 minutes

Super Secure Sentinel (S3) Design Concept

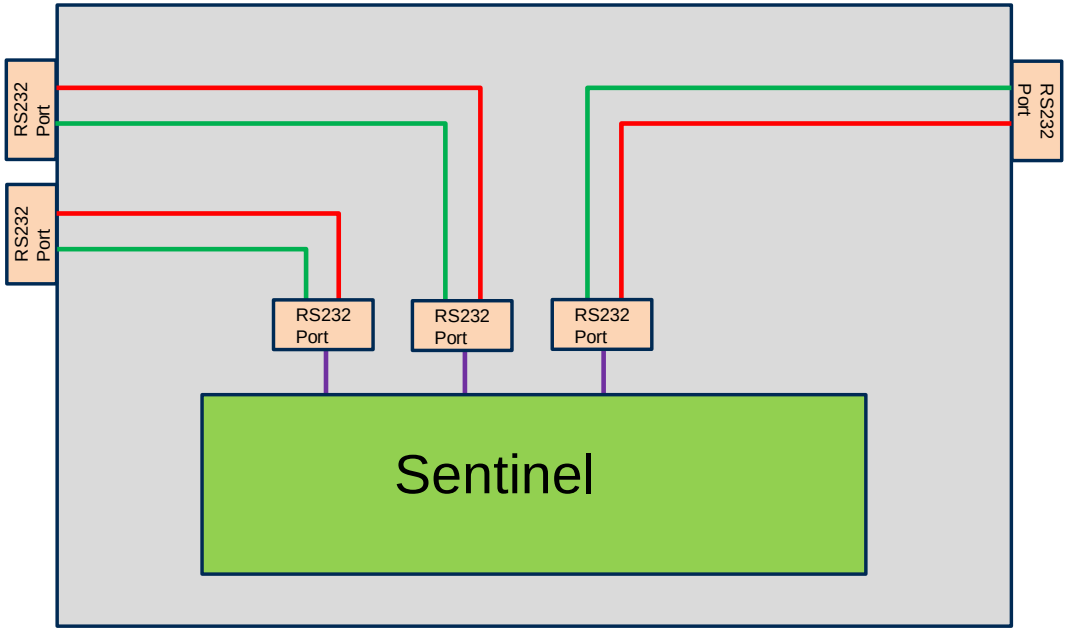
SYSTEMS ENGINEERING Research Center

High Level Architectural Overview



SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

RS232 -Based Sentinel



- Actively monitors multiple RS232 data sources (GPS Information, Password, Parameter Settings)
- Provides voting and intervention capability

30

Functional Architectural Overview

Provided by the System to be Protected

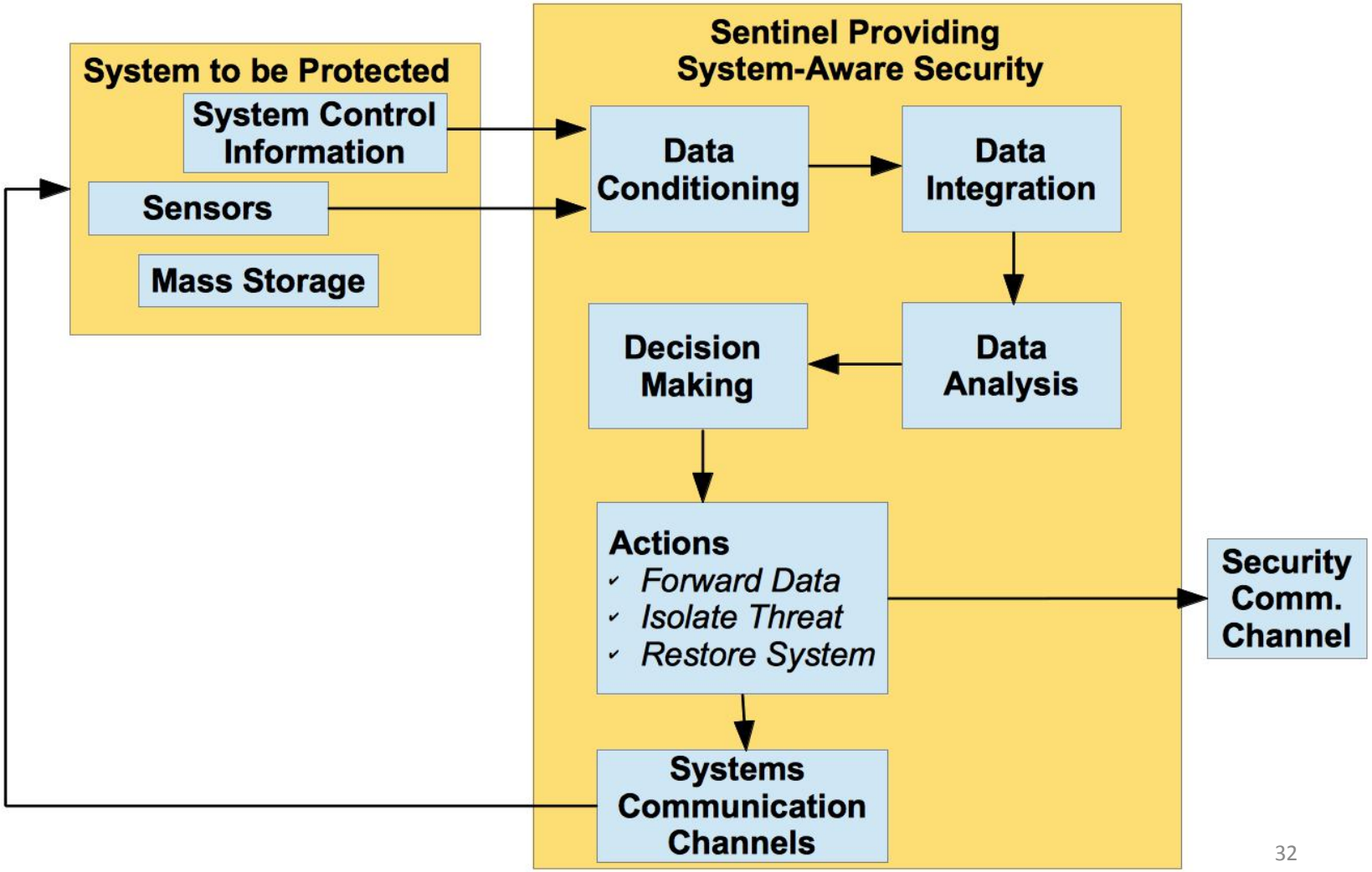
- Mass Storage as needed
- Sentinel Managed Diversely Redundant Components
  - ✓ Sensors
  - ✓ System Control Information

Sentinel Management Functions

- Verifiable Voting
- Configuration Hopping Management
- Data Consistency Checking
  - ✓ Model Based Validation
  - ✓ Confirmation of Digital Data
- Isolated Secure Communications Channel for Security Management

SYSTEMS ENGINEERING
Research Center

Sentinel Data Flow



SYSTEMS ENGINEERING
Research Center

Super Securing the Sentinel

- System-Aware Security
- Traditional Security
  - Authentication
  - Authorization
  - Access Control
  - Cryptography
- ***Sentinel Functions can be made into separable low scale implementations for application of low scale security techniques***
  - Code Validation
  - Power Signatures
  - Monitoring
  - Fingerprinting

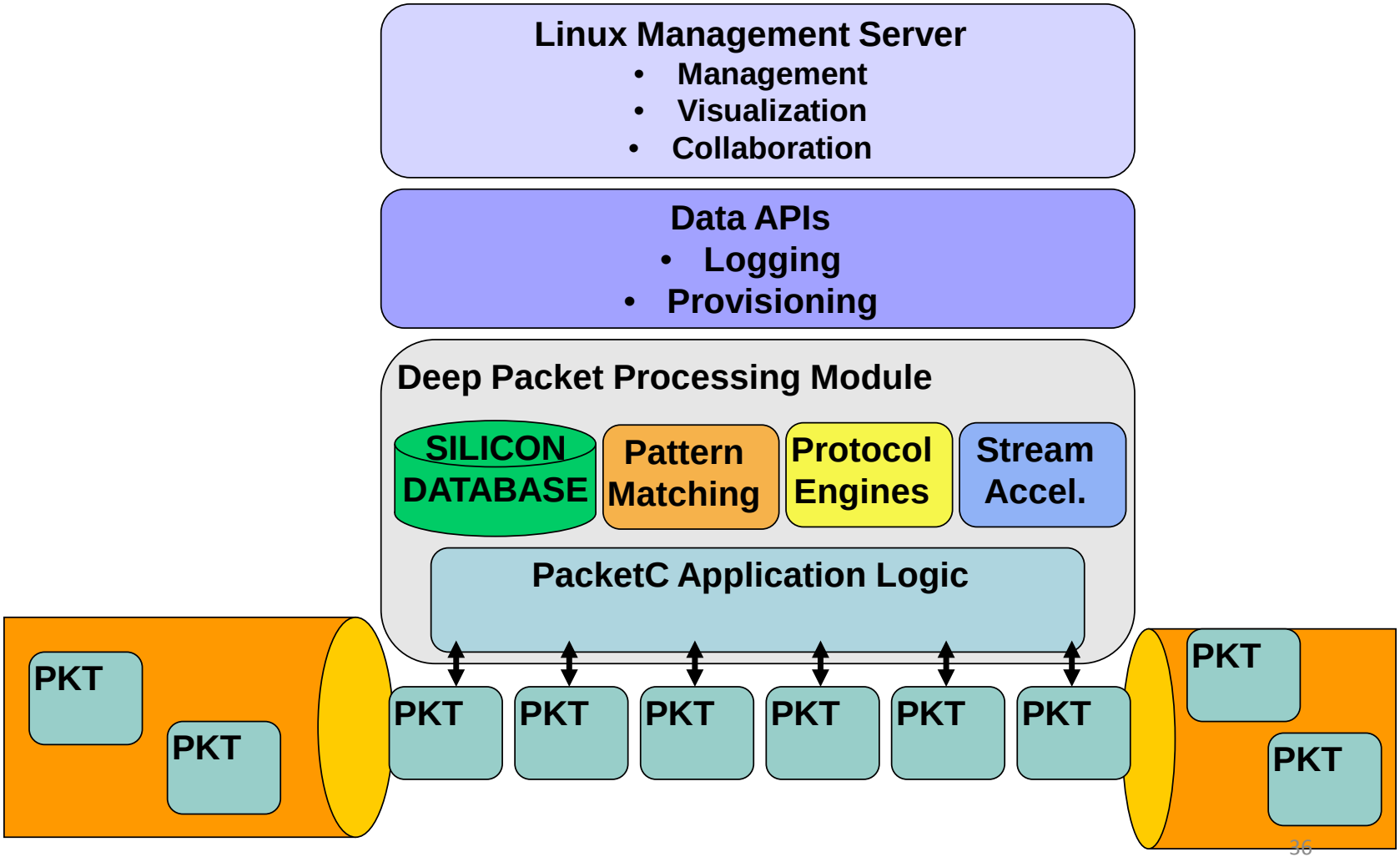
Initial Sentinel Prototype based on CloudShield Network Layer Sentinel

Sentinel Requirements Comparison

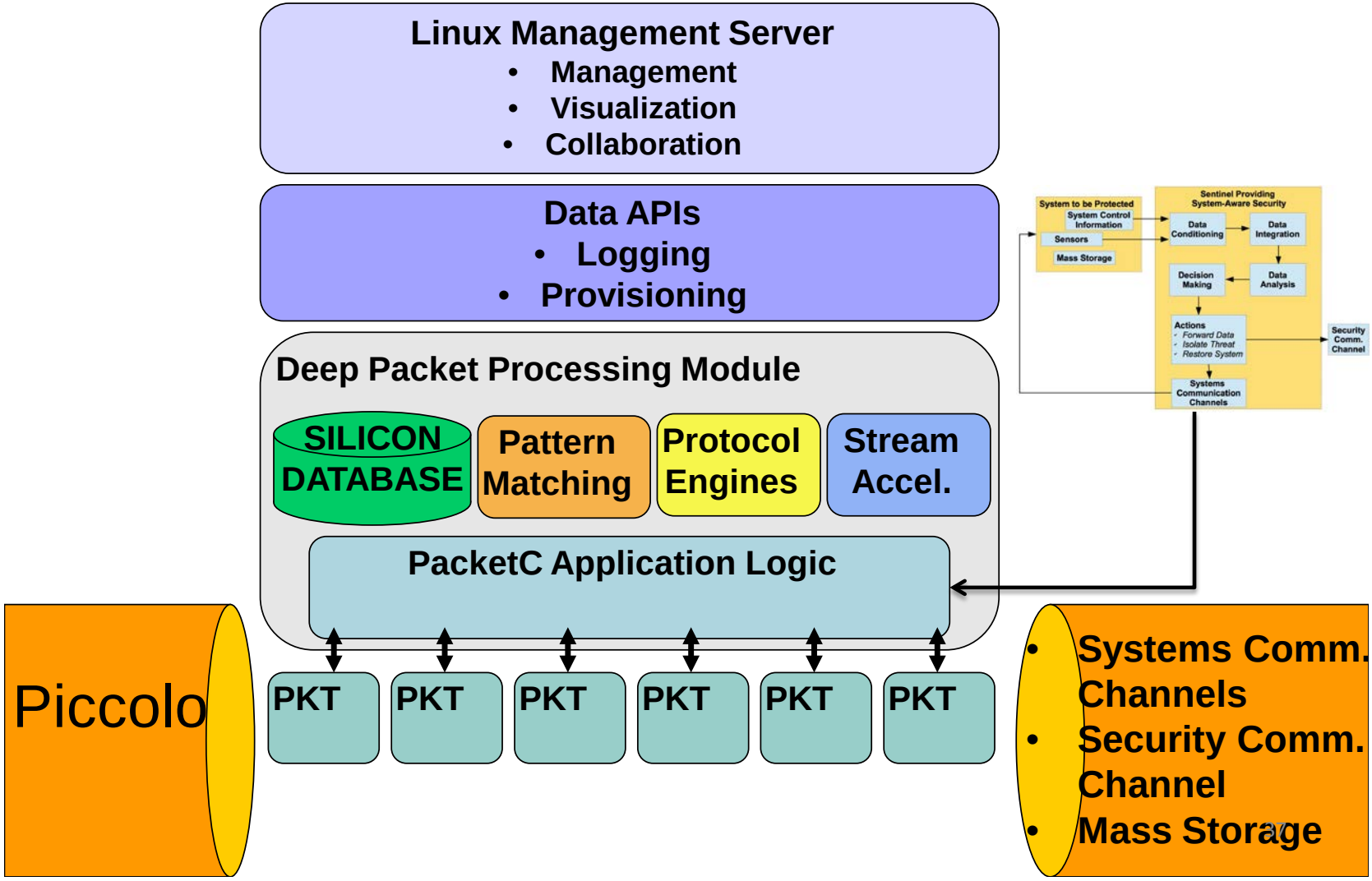
CloudShield Network Layer Design	UAV Piccolo Application	Required Workarounds
– Packet Oriented, Ethernet/IP	– Packet Oriented, RS232	– Off the shelf RS232 to Ethernet converters
– Low Latency	– Modest Latency Requirements	– None required
– Tamper Proof, Access control via one way communication	– Tamper Proof, Access control via one way communication	– None required
– 16 Input Channels, including fiber	– ~5 Input Channels	– None required
– High Throughput Rates	– Low Throughput Rates	– None required
– 17.25''W x 3.5''H x 21.0'' D	– 2 Environments • Pre-flight (Ground) • Onboard Aircraft	– Pre-flight, none required – Onboard, new HW/SW implementation
– Binary Based Design for Speed	– Binary OK for most Design Patterns	– App. specific mods as needed
– Programmable (PacketC)	– Programmable	– None required

35

CloudShield Processing Pipeline



CloudShield Processing Pipeline



SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

FY 2013 Proposal Parameters

- This proposal covers UVA/GTRI activities from March 1st – September 30th
- Two Parts– Basic and Optional
- Decision on Optional by end of March – To permit the summer employment decisions of graduate students

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Proposal (Part 1): Asymmetric Attacks

- Design and development of example asymmetric exploits for Piccolo/Mission System embedded cyber attacks
  - 4 embedded attacks (Flight disruption through parameter changes, waypoint adjustment, nav info corruption, camera gimbal control disruption)
- Schedule:
  - Basic:
    - February 28 - Design approaches and development plans completed for the 4 embedded attacks
    - September 30 – In a ground-based emulation mode, implementation and evaluation of all attacks, ***except navigation information corruption attack***
  - Option:
    - September 30 – Implementation and evaluation of an embedded navigation information corruption attack

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Proposal (Part 2): Responsive Design Patterns

- Prototype Implementation designs of responsive design patterns to the example asymmetric exploits for Piccolo/Mission System embedded cyber attacks
  - 3 data consistency-based design patterns (Parameter Assurance, Navigation Assurance/Restoration, Camera Gimbal Control Assurance)
- Schedule:
  - Basic:
    - February 28 - Design approaches and development plans completed for CloudShield implementation
    - September 30 - In a ground-based emulation mode, using CloudShield, implement and evaluate parameter assurance & gimbal control design patterns; Provide initial evaluation results for HW/SW porting options regarding live flight prototype
  - Option:
    - September 30 - In a ground-based emulation mode, using CloudShield, implement and evaluate the Navigation Assurance/Restoration design pattern

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Proposal (Part 3): Super Secure Sentinel Prototype

- Implement a highly secure Sentinel, using CloudShield as a base, which employs the selected design patterns from Part 2 of the proposal. Use System-Aware designs for the Sentinel, as well as "fingerprinting" technologies for HW and SW assurance of low scale computing modules. Start design for configuration hopping of a selected Sentinel-based design pattern.
- Schedule:
  - Basic:
    - February 28 – Selection of security features for the Super Secure Sentinel
    - September 30 - In a ground-based emulation mode, using a Super Secure CloudShield, implement and evaluate the Parameter Assurance and Gimbal Control design patterns; Provide initial evaluation results for employing configuration hopping design pattern
  - Option:
    - September 30 – Implement and evaluate the Navigation Assurance/Restoration design pattern in CloudShield. Provide initial planning and design results for porting the Sentinel functions into a flight capable HW/SW capability, including configuration hopping.

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Proposal (Part 4): Human in the Loop

- Develop the concept of employing human judgment regarding detection of attacks, to allow air-ground exchanges to guide responsive decisions, where applicable.
- Schedule:
  - Basic:
    - February 28 – Initial concept definition paper
    - September 30 – Demonstrate viable information exchange/decision support concepts for enhancing cyber security In a ground-based emulation mode, using CloudShield and a prototype implementation of the ground-based UAV control system.
  - Option:
    - September 30 - Provide initial alternatives for incorporating human-in-the-loop designs into the live flight portion of the research program.

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Proposal (Part 5): SE Architecture Decision Support Tools

- Based upon actual project application, advance the decision-support tool set for architecture selection. Initiate development of concepts for integrating geographically-separated Sentinels engaged in securing a distributed system or System-of-Systems (for UAV the ground and air sites).
- Schedule:
  - Basic:
    - February 28 – Initial concept definition paper for multi-Sentinel architectural configurations and functions, with UAV as first example.
    - September 30 – Augmentation and application of the advanced tool set to development of the plans for the live flight, air/ground Sentinel architecture

SYSTEMS ENGINEERING Research Center
A US DoD University Affiliated Research Center

Future Work Beyond UAV Project

- Pursuing other areas of application for advancing the development and use of System Aware security
- Future areas of importance include:
  - Broader set of unmanned vehicles
  - Wider set of mission applications, including geographically distributed ground-based systems and System-of-Systems
  - Big Data applications for data integrity and model integrity checking
  - Support to management of offensive cyber attack selections, by predicting possible adversary responses to US attacks