Run-time Prediction and Preemption of Configuration Attacks on Embedded Process Controllers

Lee W. Lerner, Mohammed M. Farag, Cameron D. Patterson
Cyber@VT
Bradley Department of ECE, Virginia Tech
Blacksburg, VA 24061 U.S.A.
{lwl, mmorsy, cdp}@vt.edu

ABSTRACT

Embedded electronics are widely used in cyber-physical process control systems (PCSes), which tightly integrate and coordinate computational and physical elements. PCSes have safety-critical applications, such as the supervisory control and data acquisition (SCADA) systems used in industrial control infrastructure, or the flight control systems used in commercial aircraft. Perimeter security and air gap approaches to preventing malware infiltration of PCSes are challenged by the complexity of modern networked control systems incorporating numerous heterogeneous and updatable components such as standard personal computing platforms, operating systems, and embedded configurable controllers. Global supply chains and third-party hardware components, tools, and software limit the reach of design verification techniques. As a consequence, attacks such as Stuxnet have demonstrated that these systems can be surreptitiously compromised.

We present a run-time method for process control violation prediction that can be leveraged to enhance system security against configuration attacks on embedded controllers. The prediction architecture provides a short-term projection of active controller actions by embedding an accelerated model of the controller and physical process interaction. To maintain convergence with the physical system, the predictor model state is periodically synchronized with the actual physical process state. The predictor is combined with run-time guards in a root-of-trust to detect when the predicted process state violates application specifications. Configurations can be screened before they are applied or monitored at run-time to detect subtle modifications or Trojans with complex activation triggers. Advanced notification of process control violations allows remedial actions leveraging well known, high-assurance techniques, such as temporarily switching control to a stability-preserving backup controller. Experimental simulation results are provided from a root-of-trust developed for an aircraft pitch control system.

Categories and Subject Descriptors

C.3 [**Special-purpose and Application-based Systems**]: Process control systems; C.3 [**Special-purpose and Application-based Systems**]: Real-time and embedded systems; K.6.m [**Miscellaneous**]: Security

General Terms

Design, Reliability, Security, Verification

Keywords

Configuration, embedded systems, process control systems, reliability, run-time protections, security, trust

1. INTRODUCTION

A process control system (PCS) is an embedded computer platform used to monitor and control physical processes. PCSes are a subset of cyber-physical systems, which tightly integrate and coordinate computational and physical elements. One example of a PCS is feedback control, where an embedded controller uses sensor measurements of a physical plant to compute feedback signals preserving system stability. PCSes are widely used in safety-critical infrastructure applications such as power grids, assembly lines, water systems, pipelines, power plants, and other industrial systems [1, 5]. Recent PCS attacks such as Stuxnet, which is described as the real start of cyber warfare, have highlighted embedded system vulnerabilities and the inadequacy of existing security solutions.

The Stuxnet worm infects Windows computers, spreads via networks and removable storage devices, and exploits four zero-day attacks (previously unknown vulnerabilities). Antivirus software missed the attack because programmable logic controller (PLC) rootkits hide Stuxnet modifications to the system, and two stolen certificates validate new drivers. The goal of Stuxnet is to sabotage a specific physical system by reprogramming embedded controllers to operate outside their nominal bounds by intercepting routines that read, write, and locate PLC commands and data. Many security companies state that Stuxnet is the most sophisticated attack they have ever analyzed [3], and it is estimated to have infected 50,000–100,000 computers. The primary target is believed to be the Bushehr nuclear plant in Iran, and likely caused a 15% drop in production of highly enriched uranium [4].

PCSes are usually assembled from commercial-off-the-shelf (COTS) components and third-party intellectual property

(IP). Despite the lack of trust in such components, feasible alternatives may not exist for the timely development of new systems. Trojans can be introduced into the global supply chain as either hardware or software modifications to embedded components. Since controllers are often programmable, many Trojans need only be implemented in software. Even hardware trust may be misplaced in configurable platforms, such as those utilizing field programmable gate arrays (FPGAs). PCS threats can originate from numerous sources, including hostile governments, terrorist groups, disgruntled employees, malicious intruders, and untrusted insiders.

Embedded system security solutions can be classified as either design-time or run-time techniques [16]. Design-time approaches typically seek to verify that a system is error-free pre-deployment. An example method is formal verification of system implementation to design specifications. However, system-level verification is difficult to achieve for complex assemblies of heterogeneous components. Such design-time techniques are expensive in terms of both time and effort, and can be only afforded for a limited set of applications.

Run-time techniques add trusted components to provide assurances about certain aspects of system behavior. An example of a trust anchor component is a trusted platform module (TPM) commonly used in modern personal computers [11]. Common additions include encryption and authentication modules that help assure information integrity and provide isolated compartments for applications with various levels of privilege. Such techniques can be costly in terms of design overhead and added latency and thus are often not appropriate for applications such as high-performance or general purpose computing. Yet for many applications, such as embedded cyber-physical systems, the security gains that can be achieved through trusted run-time anchors justify their presence.

We generate run-time components to simultaneously address design-for-security, -trust, and -reliability (DFSTAR). To protect against Stuxnet-like cyber threats, a secure cyber-barrier is placed around the system's control path. System behavior checks are synthesized at design-time from an application's operational and security specifications. Using this methodology, a tailored trustworthy control flow is created for the target application. This fundamentally new approach is not domain-specific and provides a proactive solution for sustaining system-level security with reliable control. Existing verification techniques are complemented but not exclusively relied upon to ensure functional system trust and security compliance. System-level PCS reliability is also addressed by incorporating specifications that should already be defined for high-reliability systems.

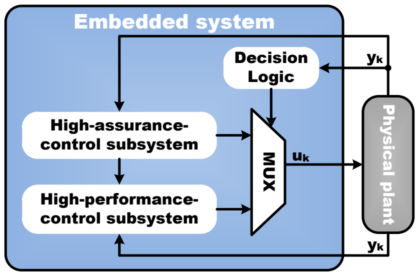
Farag et al. presented a configurable hardware-assisted application rule enforcement (CHARE) protection scheme to ensure an embedded system adheres to system specifications [8]. CHARE addresses several aspects of control security, including high-assurance module interactions and configuration programming in embedded systems. A centralized CHARE trust anchor serves as the most privileged root-of-trust for control flow, and inserts a distributed set of policy-aware specification guards on module interfaces. Specification guards provide on-line monitoring and proactive enforcement of policy rules emanating from security, performance, or reliability specifications. CHARE components tailor the hardware surrounding a system's datapath and control logic to the intended application, but do not affect the implementation of the original logic itself. A hardware-oriented solution offers resistance to software attacks and the performance necessary to implement real-time checks. CHARE reconfiguration allows for policy changes, but the trust anchor itself can only be updated with physical access or trusted channels.

The DFSTAR methodology encourages the synthesis of behavioral expectations of an entire system, including physical processes themselves. The models developed during the design stage of a cyber-physical system can be viewed as a manifestation of such system expectations. We propose that a security architecture for PCSes providing secure configuration management can be synthesized from these models following the DFSTAR methodology. In this work, a protection scheme utilizing CHARE extended with a novel process control violation prediction method is developed for embedded PCS controllers. Prediction logic incorporates a second instance of the embedded controller connected to a physical plant model running faster than real time in order to predict the future state of the physical system. The model's state is periodically synchronized with the physical plant's state to prevent state divergence. CHARE specification guards check if future system states will violate system-level policies. Controller configuration updates are tested against the current state of the process before they are applied. Additionally, if a violation is predicted after applying an update, guards switch from the faulty controller to a high-assurance, stability-preserving, backup controller until the system is stabilized. For process control networks, CHARE may be collectively applied over the network of controllers, sensors, and supervisory software.

The remainder of this paper is organized as follows: Section 2 surveys existing run-time approaches to embedded PCS reliability and security. Section 3 describes the concept of process control violation prediction and how it can be used to defend against configuration attacks on networked PCS controllers. This section also describes a prototype architecture and implementation flow for the protection system. Experimental results from applying the prediction method to a flight pitch controller simulation are provided in Section 4. Finally, conclusions and future work are summarized in Section 5.

2. EXISTING RUN-TIME APPROACHES TO PCS SECURITY AND RELIABILITY

The need for high-assurance control of physical processes by cyber-systems has led to the development of several fault detection techniques. In the case of embedded controller faults or attacks, erroneous controller behavior is ideally detected and corrected while the physical process can still return to equilibrium. PCS fault detection techniques typically observe either physical process measurements to new controller inputs or controller responses to new sensor measurements. Sha introduced a protection architecture based on monitoring physical process measurements to detect faults [14]. In this architecture, sensor measurements of the physical process are monitored by decision logic that determines if a process violation has occurred, as illustrated by Figure 1. If a violation is detected, the decision logic switches control to a high-assurance and presumably slower version of the controller until the system is stabilized. A limitation of this scheme is that system stability cannot always be recovered as the controller fault is not detected until after it has caused the physical process to deviate from allowed operational limits.



Embedded system
Decision Logic
High-assurance-control subsystem
High-performance-control subsystem
MUX
u_k
y_k
y_k
Physical plant

Figure 1: Plant fault detection [14]

Dai et al. advanced a fault detection architecture based on observing controller responses to new sensor inputs [6]. Physical process measurements are sent to both the regular high-performance version of the process controller and a trusted benchmark version of the controller algorithm. The responses of both controllers are used to generate a residual to determine if a controller fault has occurred, as shown in Figure 2. Unfortunately, the physical process is already affected by the erroneous controller output by the time the controller fault is detected and corrective actions, such as switching over to a high-assurance version of the controller, can occur. This may result in the inability to return the system to a stable state before damage is incurred.

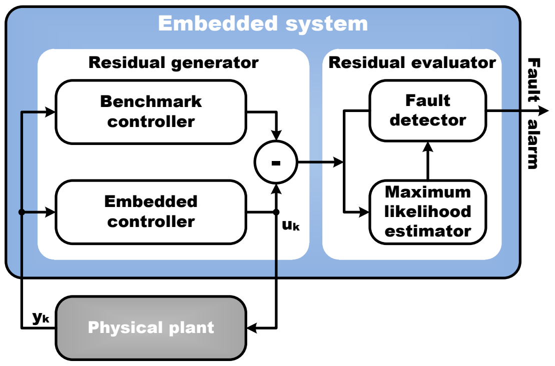


Figure 2: Controller fault detection [6]

Cárdenas et al. presented a physical model-based attack detection method with foundations in anomoly-based intrusion detection theory for computer systems and networks [3]. The specific threats addressed with this protection scheme include embedded controller intrusion attacks arising from compromised plant sensor data. Instead of using models of network traffic or software behavior, physical system models are used to develop a change detection-based intrusion detection algorithm. An embedded system implementation of a physical plant linear model runs concurrently with the plant, as illustrated in Figure 3. Controller responses are sent to both the physical plant and the embedded model. An anomaly detection module is then used to compare how sensor data measured from the physical plant compares to the response of the embedded plant model. When no differences are detected, the physical plant measurements are sent to the embedded controller which can then compute the feedback response. Sensor data intrusion is suspected when a difference is detected, in which case the embedded model's estimated physical plant state is sent to the embedded controller in an effort to filter out compromised measurements and continue plant control. This work develops a rigorous analysis and classification of PCS intrusion attacks and associated detection methods. However, it does not provide a means to detect and circumvent direct threats to controllers themselves such as configuration updates and Trojans.

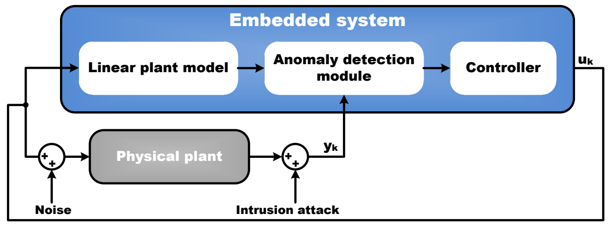


Figure 3: Controller intrusion attack detection [3]

Although many security solutions have been proposed for legacy embedded systems [2], these solutions are not optimized for process control applications. Design-time security techniques are very expensive and may not anticipate all system vulnerabilities. Such techniques often do not address vulnerabilities raised by software patches and updates, hardware reconfigurations, and zero-day exploits. This leads to a demonstrated possibility of controllers being surreptitiously compromised. An alternative approach is admitting the possibility of unanticipated threats and trying to cope with them using run-time security solutions. However, most existing run-time solutions are reactive, and can only detect erroneous controller behavior after its occurrence. Such detection methods may allow a physical processes to become unstable before corrective action can be taken. These techniques also tend to be threat-specific, leading to increased overheads for integrated solutions.

3. CONTROLLER ATTACK PREDICTION AND PREEMPTION

Our research stems from the observation that novel, deeply embedded protections are needed to cope with Stuxnet-class threats to process control systems. The specific goal of the work in this paper is to protect embedded controllers from configuration threats using a run-time security architecture synthesized with the DFSTAR methodology. Trust is neither required in the active controller-under-protection nor its update and communication infrastructure, and applies only to a small set of simple, self-contained, synthesized, and verifiable CHARE add-ons. A CHARE root-of-trust is established to ensure an application's security and reliability specifications are being observed, and essentially serves the role of an ideal control room operator. Specification guards enable the root-of-trust to directly monitor system operation and override the controller-under-protection. The DFSTAR methodology incorporates security enhancements to the system structure and automatic tool extensions to the existing design flow.

Our threat model does not distinguish between hardware faults, software bugs, and malware such as Trojans since the common denominator is non-compliant controller behavior. A Stuxnet-like threat can hide itself using sophisticated means, but is less able to disguise its ultimate goal of disturbing system stability. Regardless of how the threat originates, the role of trusted protection system is to anticipate and deter consequences to the controlled process. Based on this philosophy, we present a novel method to predict and preempt erroneous behavior in physical process control. For the PCS domain, specifications for normal system behavior are already known, and accurate models for the controlled process usually exist. Our solution is complementary to other approaches that try to validate the design or prevent malware infiltration, and serves as a last line of cyber-defense against various threats to embedded controllers.

Physical systems and processes are characterized by quantitative temporal properties such as process response time, actuator delays, and sensor time constants. These physical latencies are not inherent in system models. The vast majority of physical processes can be described and modeled as linear time invariant systems with a high degree of accuracy under very realistic assumptions. Often, a plant model running on an embedded processor can be executed much faster than a real plant operating in a physical system. In a PCS, an embedded controller responds to variations in the state of a physical plant in order to maintain system stability. Execution speed of an embedded controller corresponds to the temporal characteristics of the associated physical process. The typical operating frequency of a digital embedded system controlling a physical process is proportional to the sampling rate of the physical process. Our approach to detect erroneous behavior of embedded controllers exploits potential speed differences between a physical plant and its model, which is analogous to the difference between running a physical system and simulating it.

The main idea of our approach is examining what the controller implementation will try to do in the future by embedding a second instance of the controller with an accelerated model of the plant. The model can be implemented in either hardware (such as an FPGA) or software (perhaps on a separate processor) depending on the required speed-up. The second controller instance can be identical to the original controller and implemented on the same platform. To maintain convergence with the physical system, the model's state is periodically synchronized with the plant's state. The embedded controller instance should be subject to the same conditions as the active controller by synchronizing the model's input with the system reference input, and applying the same patches and updates to both instances. A redundant embedded subsystem incorporating these measures can accurately predict the behavior of an embedded controller for several time steps in the future.

The operation of the added protection system is illustrated in Figure 4. During regular operation, the prediction unit continuously scans projected states of the active control algorithm against the corresponding projected states of the physical process. If a fault or Trojan activity causing the physical process to deviate from allowable bounds is detected by the specification guards, the root-of-trust immediately transfers control from the active controller to the high-assurance, stability-preserving controller, such as that used in [14]. The root-of-trust is also used to interface with all new configuration and parameter updates to the active controller. When a new update is received, it is first tested in the prediction unit using the current process state as a basis. The update is rejected if the prediction unit detects a deviation from allowable process bounds during initial screening. The update is only applied to the active controller when no violations are projected. After the update has been applied, the root-of-trust resumes predictive monitoring of the active controller to deter latent attacks introduced through the new update.

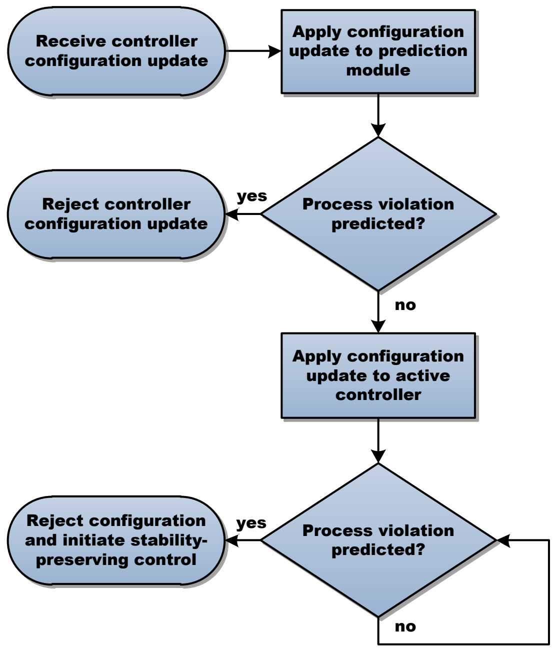

Receive controller configuration update
Apply configuration update to prediction module
Reject controller configuration update
yes
Process violation predicted?
no
Apply configuration update to active controller
Reject configuration and initiate stability-preserving control
yes
Process violation predicted?
no

Figure 4: Procedure for testing controller configurations

3.1 Prototype Controller Architecture

Some of the relevant, basic concepts of feedback control and modern control systems are presented in [7, 9]. In order to enhance the security, trust, and reliability of an embedded PCS, the existing system is augmented with a root-of-trust synthesized from a process model and specifications. As shown in Figure 5, major components of the predictive and preemptive architecture are:

- The original controller module containing an active controller-to-be-protected, a high-assurance, stability-preserving controller, and a mechanism to switch between them. This embedded system module runs at the typical sampling rate of the physical process.
- A prediction module consisting of a process model and a second instance of the active controller. This subsystem runs n times faster than the active control system module.

- A CHARE module that wraps the controller and prediction modules. This subsystem consists of specification guards as well as specialized model synchronization and timing blocks.

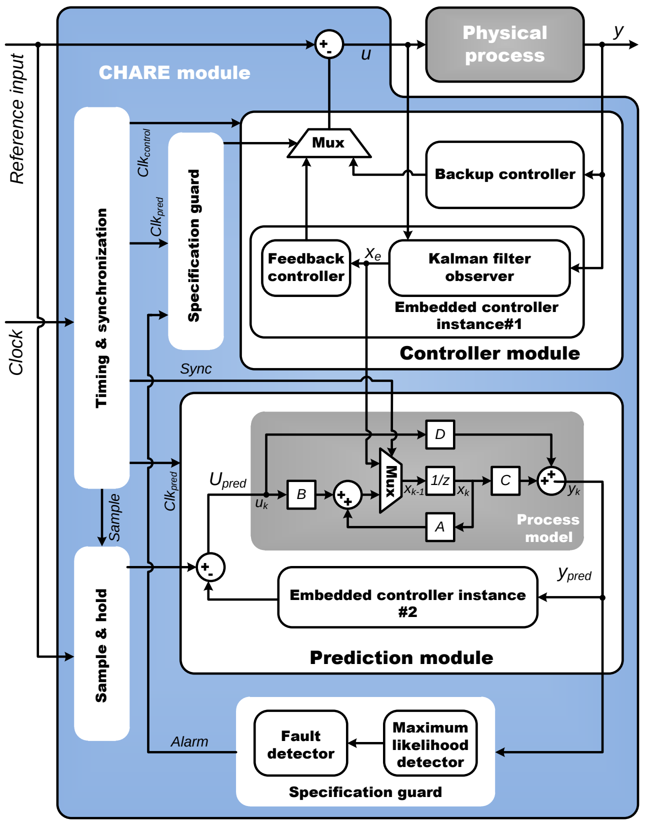


Figure 5: Predictive and preemptive security architecture

CHARE specification guards can be used to monitor either the physical process or the controller module input/output activity to assure compliance with the desired behavior of the physical process or a high-assurance benchmark controller. In this architecture specification guards monitor the process model output, as shown by Figure 5. Detection of anomalous behavior in the predictive subsystem triggers the guards to switch from the active controller of the physical process to a high-assurance controller. Recursion is possible with more than one backup controller and their predictive counterparts. The specialized synchronization unit is responsible for periodically updating the state of the model with the estimated state of the physical process. A specialized sample and hold unit updates the predictive subsystem input with the physical process reference input. The timing unit is responsible for clock generation and time emulation for the predictive subsystem.

The specification guard attached to the prediction module contains a maximum likelihood detector and a fault detector to predict faults before they actually occur in the controller-under-protection. Theoretically, if the predictive controller is secure and no threats are affecting it, the output of the controller will conform to the normal operating criteria described by the security policies embedded in the fault detector module. Any threat affecting the prediction module controller will show up later in the actual controller and, if not preempted, will increasingly affect the predictive controller's output. In practice, many factors other than those related to security threats can cause such a deviation from the normal operating conditions, such as mistuning of the controller parameters and the random noise resulting from the process. To address the challenge of distinguishing faults resulting from cyber-attacks from noise, accurate descriptions of the controller characteristics, operating conditions, and the statical distribution of noise are needed.

The fault detector does not rely on a single sample to decide the controller's integrity. For PCSEs, it has been shown that as the number of the observed samples increases, the statistical distribution of the process noise becomes Gaussian with constant mean and variance values [6]. Consequently, the statistical distribution of the controller's output follows the Gaussian noise distribution as it is the only random variable in the output equation. Deviation from the normal operating conditions caused by either cyber-threats or controller faults shifts the predictive controller's output mean and variance computed over a significant number of samples to new values outside the range defined by the security policy. In other words, deviation of the output root mean square (RMS) value from nominal bounds indicates a fault or attack in the controller-under-protection. The maximum likelihood detector unit shown in Figure 5 is used to evaluate the RMS value of the embedded model's output in the prediction module. The fault detector then tests the predictive controller's output against the the RMS value generated by the maximum likelihood detector to determine if the likelihood ratio lies within the predefined threshold range captured by the security policies.

3.2 Timing and Synchronization

Our approach advances new terminology such as: the time scaling factor n which indicates the predictive subsystem speed-up; the prediction window W_{pred} which denotes the foreseen time period; and synchronization time T_{sync} which determines the updating frequency of the model's state in the predictive subsystem. W_{pred} is function of n and T_{sync}, as shown by equation (1). T_{sync} is application-dependent whereas n is both application- and platform-dependent. Assuming flexibility in assigning n and T_{sync}, increasing n improves W_{pred} at zero cost in terms of the updating frequency, while increasing T_{sync} augments W_{pred} on the expense of reducing the updating frequency. T_{sync} is often the more flexible and tunable parameter when significant changes to W_{pred} are needed. Multiple trade-offs must be evaluated when assigning values of n and T_{sync} where the physical process characteristics and the embedded platform features are the assignment criteria.

$$W_{pred} = n \cdot T_{sync} \tag{1}$$

Time scaling is accomplished by applying modifications to both system and input signals. System modifications vary for continuous- and discrete-time models of the physical process. For a continuous-time model, time scaling is achieved by multiplying system state space matrices by the desired time scaling factor n. For discrete-time embedded systems employing digital controllers, scaling down the sampling time of the physical process by a factor of n automatically scales down the time of the system internal signals. Input signal time cannot be scaled down because this requires prior knowledge of signal contents. To tackle this problem, the model's input can be periodically synchronized with the reference input at the physical system sampling rate by assigning T_{sync} to be $T_{sampling}$ seconds. However, this approach limits the prediction window to $n \cdot T_{sampling}$ seconds.

Another approach can be adopted where the process model and the physical system are synchronized whenever the reference input to the physical system is changed. Such an approach produces an adaptive prediction window, which may not be preferred for security reasons. In a PCS, the reference input to a physical process is often the desired stable output of the system which implies that reference input changes are limited in terms of both amplitude and frequency. This implies that a sample and hold technique can be used to periodically update the model's input with the reference input without the need for an adaptive synchronization method. We adopt this approach to establish a security scheme with a controllable synchronization time and a fixed prediction window.

We consider both event-driven and time-driven faults. Accurate detection of event-driven faults depends on the proper and frequent updating of the model's state in the predictive subsystem. Figure 5 shows the switching technique created to update the model's state x_k with the estimated plant's state x_e generated by the Kalman filter state observer. The model's state updating frequency is a function of the desired prediction window and the model's input synchronization scheme. Predictive subsystem time must emulate the real time in order to successfully detect time-driven faults and properly operate time-driven modules and processes. Time emulation requires generating the predictive subsystem time in terms of n and T_{sync}, and relating it to the real time t. The predictive subsystem time is directly proportional to n, whereas T_{sync} formulates the reference time base which periodically resets the predictive time t_{pred} to the real time t as shown by equation (2). Figure 6 illustrates the predictive subsystem time for the case study described in Section 4 where the the time scaling factor n is 10, and two values of T_{sync} (1 and 10 seconds) are used for different prediction windows.

$$t_{pred} = T_{sync} \cdot \lfloor \frac{t}{T_{sync}} \rfloor + n \cdot \mathrm{mod}(\frac{t}{T_{sync}}) \qquad (2)$$

3.3 Model-based Design Flow Enhancements

Figure 7 illustrates a prototype design and implementation flow to create a root-of-trust for an embedded controller-under-protection. We focus on model-based design used to generate solutions for problem domains that have a well-established mathematical basis, such as process control, signal processing, and communications. Although model creation and analysis are a routine part of engineering, models are mostly used to explore and validate abstract solutions such as structures and algorithms, while the actual solution is implemented from scratch. In contrast, model-based design automatically synthesizes the solution—usually either hardware or software—directly from the model.

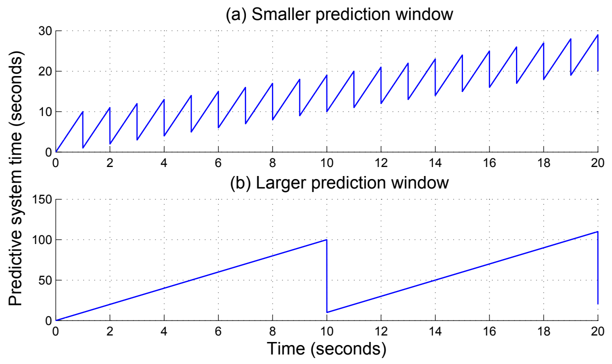


Figure 6: Relationship between real time and predictive subsystem time

Controller design begins with functional specification, often initially captured as plaintext documentation or assertions of expected behaviors of the system for the the intended application. Specifications reflecting application-specific security or reliability policies for the controller and physical process can similarly be developed at this time. Specifications guide the functional design of control algorithms in a tool such as MATLAB Simulink or Stateflow. The process model's structure is often captured graphically also using these tools in an effort to evaluate control algorithms. Tools such as Simulink Coder or HDL Coder then automate software and hardware generation of the controller, which is mapped to embedded platform components such as dedicated processors or configurable FPGA fabric.

The root-of-trust in Figure 7 consists of the process control violation prediction and CHARE modules detailed in Figure 5. The model developed to describe the process can be reused in the prediction module and implemented with the same tools used to synthesize the controller. An optimized implementation of the process model helps to reduce the time and space overheads of our predictive subsystem, which is especially important in embedded control environments that do not use PC-class hosts. As with specifications, this design flow assumes that the process model is accurate and can be trusted. Fortunately, process and high-assurance backup controller models tend to be stable, synthesized, self-contained, and subject to formal verification.

Functional and security policy specifications are inputs to the design flow for creating CHARE specification guards, as shown in Figure 7. Application-independent specification languages are used to prepare policies for synthesis, such as acceptable ranges for process sensor and controller outputs. Clearly specifying permitted ranges and rates-of-change for process and controller I/O specifically guards against Stuxnet-like attacks on processes requiring smooth control changes.

Though the DFSTAR methodology is neither hardware- nor software-specific, we choose to focus primarily on the development of root-of-trust hardware. A hardware-oriented solution provides the access and performance necessary to implement run-time protections with increased tamper resistance [12]. An example of CHARE implemented on a

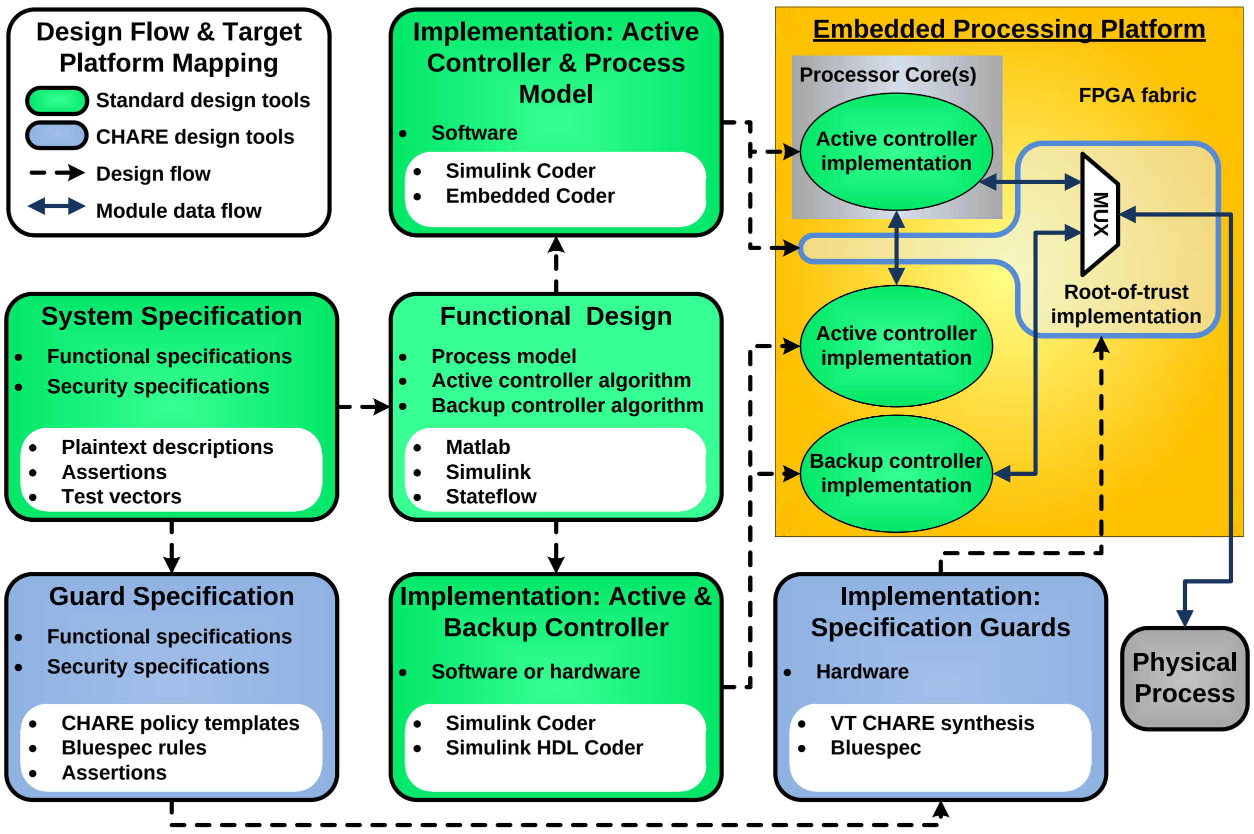


Figure 7: CHARE design flow

modern embedded processing platform marketed for industrial control applications is shown in Figure 8 [17]. The CHARE root-of-trust is synthesized to programmable hardware fabric. Processes to be monitored, such as the active controller and prediction module, can be hosted on the embedded processor cores.

The specification guard components to be validated are simple, independent, and largely synthesized from high-level abstractions, such as SystemVerilog assertions or Bluespec SystemVerilog rules with guarded atomic actions, as described in [8]. Our future work will explore methods for generating relevant assertion automata guards in detail. For the sake of verification, Simulink wrappers can be generated to enable simulation of the synthesized predictive subsystem, synchronization, and switchover blocks.

4. EXAMPLE CONTROL APPLICATION

In order to illustrate and evaluate our approach, an aircraft autopilot pitch controller is used as a case study [13]. Flight control is a safety-critical application where controller faults can have catastrophic consequences. Linear Quadratic Gaussian (LQG) control is a modern approach adopting time-domain analysis, state space representations, and state observers to enhance the control process. It concerns uncertain linear systems disturbed by additive white Gaussian noise and undergoing control subject to quadratic costs [15]. LQG controllers are widely deployed, and their structure helps to present our concepts and architecture effectively. Nevertheless, our approach is still applicable to other control techniques.

4.1 Pitch Control Process Model

The motion of an aircraft is governed by a set of six nonlinear differential equations. These equations can be decoupled into longitudinal and lateral equations under certain assumptions [13]. The pitch angle is a third-order longitudinal problem and is controlled by adjusting the angle of the rear elevator. Figure 9 shows the basic coordinate axes and forces acting on an aircraft.

As described in [13], the equations of motion of a Boeing commercial aircraft can be written as:

$$\begin{aligned}\dot{\alpha} &= -0.313\alpha + 56.7q + 0.232\delta_e\\ \dot{q} &= -0.0139\alpha - 0.426q + 0.0203\delta_e\\ \dot{\theta} &= 56.7q\end{aligned}$$

where α is the angle of attack; q is the pitch rate; θ is the pitch angle; and δ_e is the elevator deflection angle.

Using the differential equations controlling the plane motion, the state space representation of the pitch angle system is as follows:

$$\begin{bmatrix}\dot{\alpha}\\ \dot{q}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}-0.313 & 56.7 & 0\\ -0.0139 & -0.426 & 0\\ 0 & 56.7 & 0\end{bmatrix}\begin{bmatrix}\alpha\\ q\\ \theta\end{bmatrix} + \begin{bmatrix}0.232\\ 0.0203\\ 0\end{bmatrix}\begin{bmatrix}\delta_e\end{bmatrix}$$

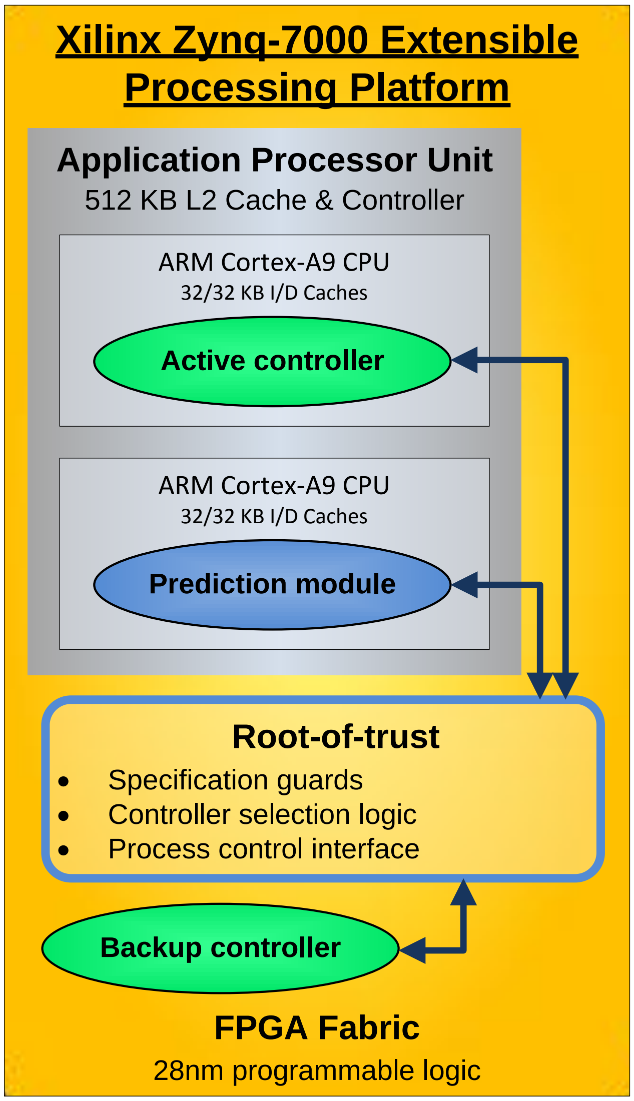


Figure 8: CHARE implemented on a Xilinx Zynq-7000 Extensible Processing Platform

$$y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} \begin{bmatrix} \delta_e \end{bmatrix}$$

In the presence of noise, this equation can be expressed in a state space form:

$$\dot{x} = Ax + Bu + \omega_{proc}$$
$$y = Cx + Du + v_{sensor}$$

where x is a column matrix composed of α, q, and θ elements representing system's state; the input u is the elevator deflection angle δ_e; the output y is the pitch angle θ; and ω_{proc} and v_{sensor} are the process and measurement noise, respectively. The noise is assumed to be Gaussian distributed with zero-mean and constant power spectral density.

The autopilot uses a feedback controller in a closed-loop configuration to stabilize the aircraft by adjusting its attitude angle. For this system, the input is the elevator deflection angle and the output is the pitch angle. LQG control is one

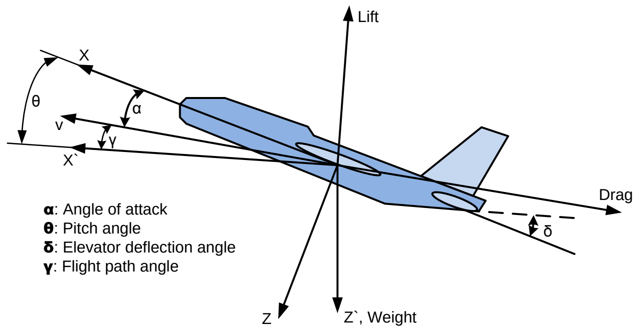


Figure 9: Coordinate axes and forces acting on an aircraft [13]

of the most commonly used optimal control techniques, and combines a Kalman filter (a linear-quadratic estimator) with a linear-quadratic regulator [15]. In the presence of noise, physical system faults can be distinguished with the aid of a Kalman filter sequence characterized by a zero mean and a fixed covariance matrix in normal operating conditions [10]. The Kalman filter optimally estimates the state of a linear system disturbed by noise. Figure 10 shows the structure of the LQG control technique.

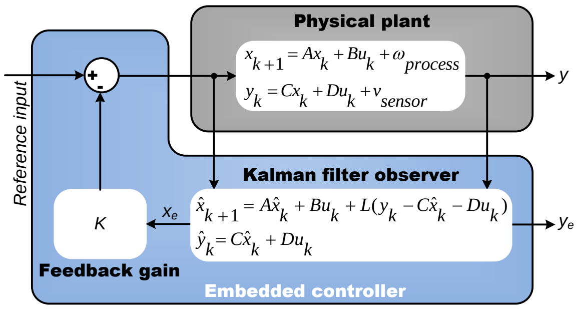

Physical plant
$x_{k+1} = Ax_k + Bu_k + \omega_{process}$
$y_k = Cx_k + Du_k + v_{sensor}$
y
Reference input
+
-
Kalman filter observer
$\hat{x}_{k+1} = A\hat{x}_k + Bu_k + L(y_k - C\hat{x}_k - Du_k)$
$\hat{y}_k = C\hat{x}_k + Du_k$
x_e
y_e
K
Feedback gain
Embedded controller

Figure 10: LQG control architecture

Figure 11 illustrates the step response of the pitch controller system for both the open-loop system and the closed-loop feedback control configurations. Knowing the system state helps maintain synchronization between the physical system and our predictive subsystem. The design meets or exceeds the constraints with a maximum overshoot of less than 10 percent, a rise time of approximately 2 seconds, a settling time of less than 10 seconds, and a steady-state error of less than 2 percent.

4.2 Preliminary Results and Evaluation

Simulink is used to model and evaluate the application and security enhancements. In our experiments, the sensor sampling rate is 100 samples/sec, the time scaling factor n is 10, and two values of T_{sync} (1 and 10 seconds) are used for different prediction windows. Figure 12 illustrates the step response of a stable pitch control system versus the predictive subsystem, and synchronizes between both systems for the test case of $T_{sync} = 1$ second. A moving window allows periodic projection of the future system state from the updated current system state.

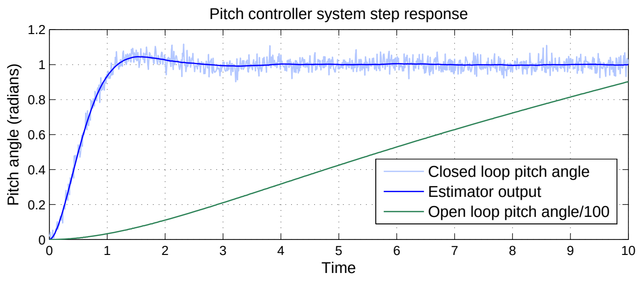


Figure 11: Open- and closed-loop step response of the pitch angle control system

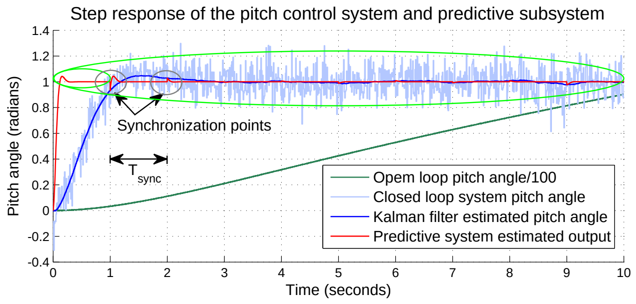


Figure 12: Step response of a stable pitch control system versus predictive subsystem

To illustrate the effectiveness of our approach in predicting and preempting erroneous behavior, we insert a time-driven fault in the embedded controller's model. The fault is injected by manipulating a single element of the gain matrix in the feedback controller model. This fault is kept dormant for a pre-assigned period of time as shown by Figure 13(a). Such a fault can be inserted during aircraft maintenance as a system patch or update to a rigorously verified embedded controller, and can force the plane out of its stable state in a very short time.

Figure 13(b) and (d) illustrate the predictive subsystem output for different windows. Initial system stability is assumed. The smaller prediction window predicts the fault 10 seconds before its occurrence, and the larger prediction window foresees the fault about 50 seconds before its occurrence. Smaller windows are more accurate but see the fault later. As time advances, the predictive subsystem anticipates the fault as the sawtooth waveform with increasing peaks indicating fault advancement. The backup controller used in this evaluation is simply the developed fault-free LQG controller. Figure 13(c) and (e) show the multiplexer selection signal and the time of switching for the two selected prediction windows. For both case studies, the countermeasure has successfully prevented faults occurring as illustrated by Figure 13(f).

This proof-of-concept experiment was performed by modeling the process and our protection system in Simulink. Future experiments will examine a fully hardware-oriented root-of-trust. Simulink profiler results for the pitch controller model are given in Figure 14. One minute of real time is simulated in 22 sec of CPU time on a single core of a 2.80

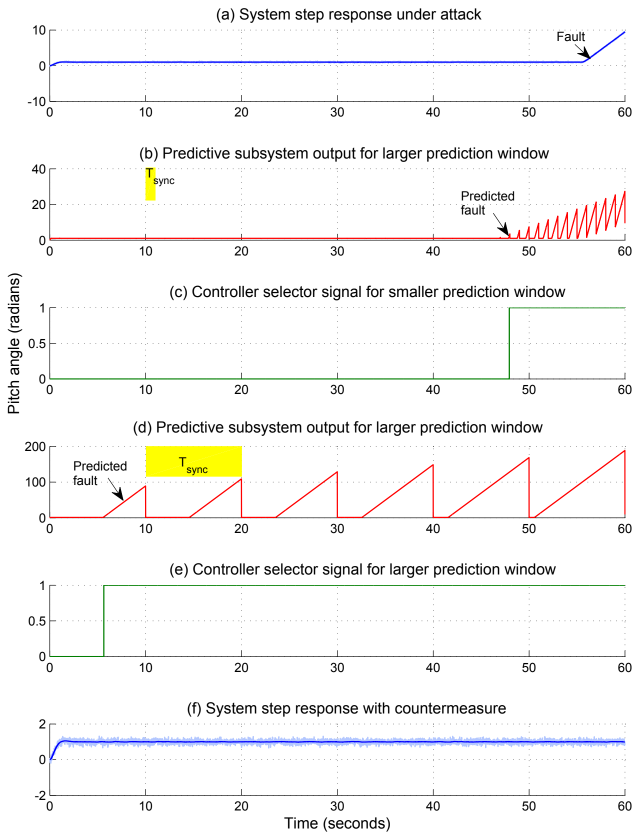


Figure 13: Step response of a pitch control system and the predictive subsystem under attack

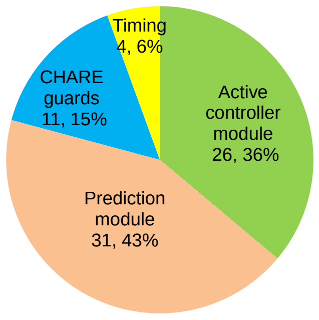
GHz Intel Core i7 workstation with 24GB of memory and running the Linux 2.6.32 kernel. In practice, optimized C implementations would run more quickly than the Simulink models.

The predictive subsystem and active control system design complexity are almost the same, and the CHARE trust anchor contributes about 25% of the total complexity as measure in terms of total software methods. However, it is anticipated that the overhead of the root-of-trust will shrink significantly for a hardware-based implementation. Most of the CPU time is consumed by the predictive subsystem as it runs n times faster than the active controller. On a multicore platform, however, the predictive subsystem runs concurrently with the active controller.

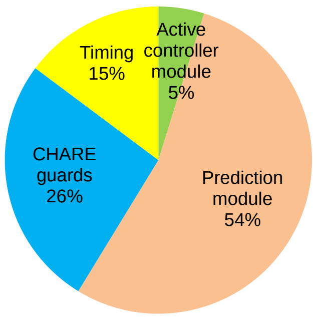
5. CONCLUSIONS AND FUTURE WORK

Comprehensive rather than point solutions are needed to help PCS infrastructure withstand an emerging malware onslaught. As illustrated by Stuxnet, preventing malware infiltration is difficult in complex, networked control systems having zero-day exploits. Trojans may also arise from the global supply chain and use of third-party IP. This leads to a demonstrated possibility of controllers being surreptitiously compromised. Erroneous controller behavior must be detected before it critically affects a physical process.

Existing solutions to run-time bug and fault detection in-



(a) Number of methods



(b) Execution time

Figure 14: Pitch controller static / dynamic analysis

clude monitoring the process state arising from past controller actions, or comparing present outputs from independent controllers. Our run-time system includes a second instance of the active controller connected to a model of the plant giving a short-term projection of future controller actions and process state. The model's state is periodically synchronized with the plant's state to prevent divergence. Erroneous controller behavior is detected before it affects the physical process, allowing preemptive alarms or actions.

The blocks conferred with trust should be minimal in complexity and number so that synthesis and formal verification methods may be applied. In addition, these blocks should have rigorous update procedures, and use distinct software and hardware resources. Ideally the trusted blocks do more than just protect against malware by also enhancing reliability in the presence of software bugs and hardware faults. The DFSTAR methodology meets these goals by exploiting: (1) the inherent controllability and observability of a PCS; (2) the existence of specifications for normal PCS operation; and (3) the model-based design approach commonly used during PCS development.

Tools are under development to automatically generate the trusted components described in this paper. FPGA platforms enable the use of integrated yet independent resources for monitoring functions, and allow both hardware and software implementation of synthesized blocks. Once the tools are complete, we will be able to assess the design-time and run-time overheads of the DFSTAR methodology. After targeting simple controller applications such as pitch control, we will look at inserting a system monitor in a network of process controllers. We also plan to see how these tools scale up to to more complex multiple-input-multiple-output (MIMO) controllers used in modern PCS platforms.

6. ACKNOWLEDGMENTS

The CHARE framework used in this work was developed in the AUSTIN (An Initiative to Assure Software Radios have Trusted Interactions) project supported by the National Science Foundation, Grant No. CNS-0910557.

7. REFERENCES

[1] M. Brundle and M. Naedele. Security for process control systems: An overview. *Security Privacy, IEEE*, 6(6):24–29, Nov–Dec 2008.

[2] A. Cárdenas, S. Amin, and S. Sastry. Secure control: Towards survivable cyber-physical systems. In *Distributed Computing Systems Workshops, 2008. ICDCS '08. 28th International Conference on*, pages 495–500, Jun 2008.

[3] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS'11, pages 355–366, 2011.

[4] T. Chen. Stuxnet, the real start of cyber warfare? [editor's note]. *Network, IEEE*, 24(6):2–3, Dec 2010.

[5] F. Cohen. Automated control system security. *Security Privacy, IEEE*, 8(5):62–63, Sep–Oct 2010.

[6] C. Dai, S. Yang, and L. Tan. An approach for controller fault detection. In *Fifth World Conference on Intelligent Control and Automation (WCICA)*, volume 2, pages 1637–1641, Jun 2004.

[7] R. C. Dorf and R. H. Bishop. *Modern Control Systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 9th edition, 2000.

[8] M. M. Farag, L. W. Lerner, and C. D. Patterson. Thwarting software attacks on data-intensive platforms with configurable hardware-assisted application rule enforcement. In *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, pages 207–212, Sep 2011.

[9] G. F. Franklin, D. J. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 4th edition, 2001.

[10] C. Hajiyev and F. Caliskan. Fault detection in flight control systems via innovation sequence of Kalman filter. In *Control '98. UKACC International Conference on*, pages 1528–1533 vol.2, Sep 1998.

[11] S. L. Kinney. *Trusted Platform Module Basics: Using TPM in Embedded Systems (Embedded Technology)*. Newnes, 2006.

[12] L. W. Lerner, M. M. Farag, and C. D. Patterson. Interacting with hardware Trojans over a network. In *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*, June 2012.

[13] W. Messner and D. Tilbury. *Control tutorials for MATLAB and Simulink: User's Guide*. Addison-Wesley, 1998.

[14] L. Sha. Using simplicity to control complexity. *Software, IEEE*, 18(4):20–28, Jul–Aug 2001.

[15] L. Surhone, M. Tennoe, and S. Henssonow. *Optimal Projection Equations*. VDM Verlag Dr. Mueller AG & Co. Kg, 2010.

[16] M. Tehranipoor and F. Koushanfar. A survey of hardware trojan taxonomy and detection. *Design Test of Computers, IEEE*, 27(1):10–25, Jan–Feb 2010.

[17] Xilinx, Inc. *Zynq-7000 EPP Overview, DS190 (v1.1.1)*, June 2012.