Formal Verification for Mission Assurance in Cyberspace: Education, Tools, and Results

Shiu-Kai Chin[◇♡], Erich Devendorf[♡], Sarah Muccio[♣], Susan Older[◇♡], and James Royer[◇]

Abstract— **Mission assurance is the assurance of the correctness, integrity, security, and availability of critical capabilities necessary to complete a mission successfully. National security depends on the integrity of command and control for military systems, the power grid, and financial systems. Thus, the alarming lack of personnel capable of doing mathematically rigorous specification, design, verification, testing, and procurement of trustworthy systems is a national weakness with profound implications for national security. This paper reports the results of a pilot program at the undergraduate level whose objectives include equipping undergraduate computer engineers and computer scientists with the theory, methods, and tools necessary for formal specification and verification of mission-essential functions in cyberspace.**

Index Terms— **Cyber security, formal verification, access control, theorem proving, undergraduate education**

I. Introduction

"No operator should ever have to ask ... 'Will my weapon work?' ... Cyberspace warfare creates just this possibility." – Gen John Shaud, USAF

The United States' power, electrical grid, financial services, and other critical infrastructure are inextricably intertwined with cyber operations that depend on computer-enabled command and control systems. The threat to national security by compromise of these systems' integrity is well publicized. Furthermore, the desirability and strategic importance of rigorous, certifiable, reusable, and replicable assurance using formal verification is widely acknowledged.

Despite these facts, new systems are often designed by the newest engineers, for understandable economic reasons. Without a solid educational foundation that supports skills in rigorous specification, design, and verification, there is little hope for improved assurance of system integrity and security. This capability is generally believed to be beyond the grasp of practitioners and computer engineering and science undergraduates.

Our results disprove this belief. Our conclusions are based on a ten-year partnership and experimentation to devise the theory, practice, techniques, tools, and means for technology transfer (i.e., education and courses at the undergraduate level) necessary for assuring the integrity and security of critical systems. Our efforts are focused on the next generation of leaders in cyberspace: undergraduate computer engineers, computer scientists, and reserve officer training corps (ROTC) cadets. Our conclusions are based on the achievements of more than 265 students from more than 50 US universities who have gone through our programs. This paper gives an overview of the theory, application, tools, and courses we use to equip future leaders to think rigorously about mission assurance in cyberspace.

The rest of this paper is organized as follows. Section II presents the context of our work in terms of our educational, research, and internship programs. Section III describes our methods in terms of conceptual roots, objectives, and tools. We present our results in Section IV and our lessons learned in Section V. Our conclusions are in Section VI.

II. Context: ACE-CS, Internships, and the Cyber Engineering Semester

Our results are based on a ten-year educational partnership among academia, industry, and government. Government, industrial, and academic researchers work together to educate and mentor students to solve real-world problems. Team members do not have fixed roles: government and industrial staff at times teach theoretical courses, while academic faculty often teach tools applied to specific problems. In all cases, a deliberate attempt is made to integrate theory with practice to solve real problems.

A. ACE-CS: Cyber Security Boot Camp

This partnership began with the 2003–2010 Air Force Research Laboratory (AFRL) Advanced Course in Engineering (ACE-CS) Cyber Security Boot Camp [1], [2], a ten-week summer program based on the tenets of the General Electric Advanced Course in Engineering (GE-ACE) [3].

The GE-ACE and ACE-CS programs emphasize mathematically rigorous solutions to real engineering problems. Both build a culture that values teamwork, rigorous analysis, and on-time performance. Programs like GE-ACE and ACE-CS naturally lend themselves to building a cadre of professionals with shared values. People who successfully complete ACE-CS have immediate credibility with one another and with the senior officers who actively seek out ACE-CS graduates. Over the last ten years, the AFRL ACE-CS program has

[◇]Electrical Engineering & Computer Science, Syracuse University
[♣]Air Force Research Laboratory Information Directorate, Rome, NY
[♡]Serco-NA, Rome, NY

Distribution Statement A - Approved for Public Release - Distribution Unlimited Document #88ABW-2012-1812, dated 29 MAR 2012

graduated 226 ROTC cadets and civilians from over 40 US universities. Many ACE-CS graduates are currently on active duty and distinguishing themselves by their capabilities and achievements.

B. Information Assurance Internships

In 2011, ACE-CS successfully transitioned to the Air Force Institute of Technology (AFIT-ACE). This transfer enabled the AFRL ACE-CS team to launch the AFRL Information Assurance (IA) Internship Program in the summer of 2011. When compared to ACE-CS, the IA Internship Program has a greater emphasis on research and problem solving throughout the summer. Whereas ACE-CS followed the GE-ACE model of a new problem each week, the IA Internship Program elected to focus on an overall research problem and develop the rigorous theoretical and practical knowledge needed to solve it. The Summer 2011 research problem focused on the science of mission assurance in a cloud-computing environment, with an emphasis on Air Force mission-essential functions in a contested environment.

The 2011 IA Internship had 13 students. The 2012 IA Internship will have a total of 25 students from universities across the US, including Massachusetts Institute of Technology, University of Illinois at Urbana-Champaign, Rochester Institute of Technology, Michigan Technological University, University of Texas at El Paso, Southern University, Missouri University of Science and Technology, Bethel College, Rutgers University, University of Maryland, University of Virginia, Kennesaw State University, Iowa State University, Syracuse University, Rose Hulman University, University of Delaware, Clarkson University, Texas A&M University, Ithaca College, University of Dayton, Rennsalaer Polytechnic Institute, University of Montana, University of Michigan, and University of Pittsburgh.

C. The Cyber Engineering Semester

The same core team that planned and executed ACE-CS and the IA Internship Program also planned and executed the *Cyber Engineering Semester* (CES). By cyber engineering, we mean the computer engineering and computer science necessary for *engineering trustworthy systems operating in contested environments*. The pilot program was executed successfully in the Fall 2011 semester with six undergraduates (five juniors and one senior) from Michigan Technical University, Syracuse University, and Texas A&M University. Three students were Air Force ROTC cadets, and the rest were civilian students.

Figure 1 shows the Fall 2011 Cyber Engineering schedule. The 18-credit pilot program consisted of five courses and a paid internship at the Air Force Research Lab (AFRL) Information Directorate. Each course is briefly described in Figure 2.

As in the ACE-CS and IA Internship programs, the CES by design keeps the students together as a cohort. The faculty and staff meet regularly to “connect the dots” for the students (i.e., enable the students to see how theory, concepts, and tools could be integrated to solve real-world problems seen in class projects and the AFRL internship). The same ACE-CS cultural values are inculcated: teamwork, rigorous analysis, and on-time performance. Alumni from ACE-CS, the IA Internship Program, and the CES are part of a growing cohort of leaders in cyberspace.

	Monday	**Tuesday**	**Wednesday**	**Thursday**	**Friday**
0800	Secure Hardware Lab	Secure OS	Secure Hardware Lab	Secure OS	Paid Internship at Air Force Research Lab
0900					
1000					
1100					
1200	Secure Architecture	Cyber Engineering Seminar	Secure Architecture	Cyber Engineering Seminar	
1300					
1400	Engineering Assurance Lab		Engineering Assurance Lab		
1500					
1600					

Fig. 1: Cyber Engineering schedule for Fall 2011 semester

1) **Cyber Engineering Seminar (3 credit hours):** Exploration of the operational art of cyber operations at the strategic, operational, and tactical levels through critical thinking exercises, problem solving, writing, and presentation
2) **Engineering Assurance Lab (4 credit hours):** Introduction to the theory, practice, and tools for building and verifying systems
3) **Design of Secure Operating Systems (4 credit hours):** Design and implementation of modern secure operating systems. Authentication, authorization, and access control
4) **Secure Computer Architecture (4 credit hours):** Introduction to fundamentals of computer architecture with emphasis on security measures in hardware implementation of processor units
5) **Secure Hardware Engineering Lab (3 credit hours):** Design and implementation of hardware to preserve both the confidentiality and integrity of data, authenticate keys, people, and privileges, and protect physical memory

Fig. 2: Cyber Engineering Semester course descriptions

All involved describe the CES as *“intense.”* Because of space limitations, in this paper we focus on the formal reasoning and verification elements of the CES, which are perhaps the most novel aspects of the program. Teaching security and formal verification to undergraduates using theorem provers is rare, if not considered virtually impossible by most academics.

III. Methods: Objectives, Central Concept, and Power Tools

The three programs—ACE-CS, IA Internship Program, and CES—have their programmatic and philosophical roots firmly planted in the GE-ACE. Founded in 1923, the GE-ACE was created to address a particular need as described by Francis Pratt, then GE vice president of engineering [3]: *“[A] noticeable number of our most accomplished theoretical engineers and laboratorians have either pursued post-graduate studies at European universities or else have had all of their scholastic training abroad, and the time is approaching when their successors will have to be found”*. Simply put, GE in 1923 was facing a critical shortage of engineering leaders necessary for its future survival.

Robert E. Doherty, who later went on to be president of Carnegie Institute of Technology, founded the (still extant) GE-ACE to solve Pratt’s problem. His solution was the three-year

Level	Cyber Engineering Seminar	Engineering Assurance Lab
Comprehension	- Realize the operational context of cyber engineering - Describe the steps to access local and remote systems - Explain security as part of software development - Describe the information life cycle	- Explain the meaning and discuss the interpretation of given mathematical or logical expressions - Restate in the Haskell functional language or Higher Order Logic (HOL) theorem prover a given mathematical or logical definition or property
Application	- Circumvent OS authentication mechanisms to gain access - Harden BIOS, boot loaders, OS, and applications - Eliminate code-level I/O-derived vulnerabilities	- Use Haskell to create an executable specification - Use HOL to restate a specification in higher-order logic - Restate the syntax and operational semantics of a language in Haskell and HOL - Use HOL to define concepts of operations (CONOPS), access-control descriptions, or policies
Analysis	- When given a mission, determine the requirements for mission assurance - Apply knowledge of past revolutions in military affairs to current situations	- Use the rules of structural operational semantics (SOS) or access-control logic (ACL) to prove properties in SOS or ACL - Use QuickCheck or HOL to check or verify properties
Synthesis	- Derive actionable intelligence on a given target from open source resources - Alter OS and application software to maintain access and obscure activities - Create operational deceive, disrupt, degrade, deny, and destroy effects	- Construct in Haskell or HOL, appropriate data types, predicates, functions, semantics rules, and proofs for specifications, definitions, or properties - Devise specialized inference in HOL for reasoning about access control, specific data structures, and concepts of operation
Evaluation	- Assess software applications for input-derived vulnerabilities	- Assess the correctness of proofs using the rules of SOS, access-control logic, or HOL

Fig. 3: Educational outcomes listed in increasing level of sophistication

in-house GE-ACE program, designed to develop the following attributes [3]:

1) *the ability to identify and solve real engineering problems,*
2) *concern for the readers of engineering reports,*
3) *generalists with competence is a wide variety of engineering fields,*
4) *an understanding of the use and misuse of mathematical analysis, and other ways of solving engineering problems, and*
5) *the realization that the engineer's primary purpose is not mathematical virtuosity, but the improvement of methods and products.*

Eighty-nine years later, these attributes remain central to the IA Internship and CES programs. Although the IA Internship and CES Programs are less than three years in duration, what truly distinguishes them from the GE-ACE is the fact that they are operated jointly by government, industry, and academia. This partnership affords a degree of mutual support and agility that programs operated solely by government, industry, or academia cannot match.

A. Objectives

Figure 3 lists the educational outcomes for the Cyber Engineering Seminar and the Engineering Assurance Laboratory. The Cyber Engineering Seminar course was taught by AFRL staff and industry staff. The overall objectives of the seminar course are similar to the GE-ACE: *rigorous solution of real engineering problems in a professional environment*. Students are required to do a formal mathematical analysis, write a full engineering report, and make professional presentations defending their solutions.

The Engineering Assurance Laboratory (EAL) was taught by university faculty and focused on formal-verification tools. This course is unusual at the undergraduate level due to its use of functional programming (Haskell) and theorem provers (HOL). Functional programming (which emphasizes the mathematical relationships between functions' input and output) and theorem provers (which check the correctness of proofs) both serve the same end: *rigorous mathematical assurance of correctness*. The importance of computer-assisted reasoning tools in the context of mission assurance and security cannot be overstated: the computer assistance is necessary both for verification and for the credible dissemination of results. The situation is analogous to hardware design, which today depends on computer-aided design (CAD) tools such as design-rule checkers, logic-to-layout verifiers, and model checkers.

The EAL exists to support the problem solving required in the Cyber Engineering Seminar and the AFRL internship. The first half of the EAL introduces the basics of Haskell programming and HOL theorem proving, along with structural operational semantics. The combination of functional programming and HOL theorem proving is of great practical value, because HOL (like many other theorem provers) is a collection of functional programs whose inference rules are functions that return objects of type *theorem*. In the second half of the EAL, EAL homework problems are directly related to the types of problems seen in the Seminar and Internship.

B. A Command-and-Control Calculus

We view the integrity of command and control to be of paramount importance for mission assurance. In the same way that propositional logic is the basis of digital design for computer hardware engineers, a calculus for command and control is central to designing and verifying the logical consistency of mission command and control, protocols, and policies for security and integrity.

The command-and-control calculus we use is the access-control logic described in the textbook *Access Control, Security, and Trust: A Logical Approach* [4]. This calculus is

$$\textit{Says}\quad \frac{\varphi}{P \text{ says } \varphi}$$

$$P \text{ controls } \varphi \stackrel{\text{def}}{=} (P \text{ says } \varphi) \supset \varphi$$

$$P \text{ reps } Q \text{ on } \varphi \stackrel{\text{def}}{=} P \mid Q \text{ says } \varphi \supset Q \text{ says } \varphi$$

$$\textit{Controls}\quad \frac{P \text{ controls } \varphi \quad P \text{ says } \varphi}{\varphi} \qquad \textit{Derived Speaks For}\quad \frac{P \Rightarrow Q \quad P \text{ says } \varphi}{Q \text{ says } \varphi}$$

$$\textit{Reps}\quad \frac{Q \text{ controls } \varphi \quad P \text{ reps } Q \text{ on } \varphi \quad P \mid Q \text{ says } \varphi}{\varphi}$$

$$\textit{\& Says (1)}\quad \frac{P \,\&\, Q \text{ says } \varphi}{P \text{ says } \varphi \wedge Q \text{ says } \varphi} \qquad \textit{\& Says (2)}\quad \frac{P \text{ says } \varphi \wedge Q \text{ says } \varphi}{P \,\&\, Q \text{ says } \varphi}$$

$$\textit{Quoting (1)}\quad \frac{P \mid Q \text{ says } \varphi}{P \text{ says } Q \text{ says } \varphi} \qquad \textit{Quoting (2)}\quad \frac{P \text{ says } Q \text{ says } \varphi}{P \mid Q \text{ says } \varphi}$$

$$\textit{Idempotency of} \Rightarrow\quad \frac{}{P \Rightarrow P} \qquad \textit{Monotonicity of} \mid\quad \frac{P' \Rightarrow P \quad Q' \Rightarrow Q}{P' \mid Q' \Rightarrow P \mid Q}$$

Fig. 4: Sample inference rules of the access-control logic

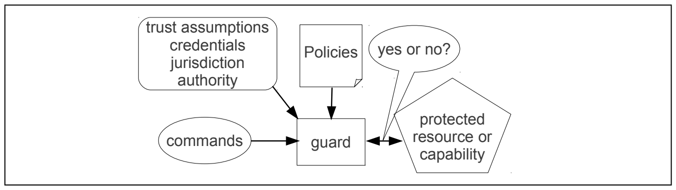


Fig. 5: Basis for determinning whether to grant access request

based on an access-control and authentication logic originally defined by Lampson, Abadi, Burrows, Wobber, and Plotkin [5], [6]. The inference rules of the command-and-control calculus (some of which appear in Figure 4) are guaranteed to be logically sound with respect to their (Kripke) semantics.

The central question addressed by the calculus is illustrated in Figure 5. Given a request (or command) to access a protected resource or capability, how do we *logically derive* whether or not to grant that request based upon specific policies, trust assumptions, credentials presented, and assumptions about authorities and their jurisdiction? This question is analogous to what computer hardware engineers ask when they use logic to derive the output values of computer hardware.

The command-and-control calculus can be applied at all levels of abstraction, from the control of physical memory and network protocols, up to and including the security and integrity policies of organizations. Using the same command-and-control calculus across multiple levels of abstraction enables consistent interpretations and implementations of policies.

A calculus is useful to the extent that it describes useful concepts and relationships. Figure 6 shows how some important relationships are represented in the logic. To illustrate how the calculus works, we show how to derive conclusions using the relationships in Figure 6 and the inference rules in Figure 4.

As a simple example, suppose that Alice's public key is K_a and that this association is certified by certificate authority CA whose key is K_{ca}. This certification is a *public-key certificate signed by* K_{ca}. Further suppose that Bob trusts or recognizes CA's authority over public-key certificates and that Bob believes K_{ca} is CA's public key. The proof in Figure 7 provides a formal justification for Bob to conclude that K_a is Alice's key (lines 1 through 3 indicate Bob's starting assumptions, and everything afterward follows as the result of inference rules from Figure 4.)

As another example, suppose that Alice is Blue Force Commander (BFC) and in that role has the authority to issue the *go* command to start a mission. Suppose further that Bob is in Alice's chain of command and that he receives Alice's *go* command encrypted with key K_a. The proof in Figure 8 provides a formal justification of Bob's conclusion that the encrypted order he receives is legitimate and actionable.

The proofs in Figures 7 and 8 give rise to the two new *derived inference rules* shown in Figure 9, which are guaranteed to be sound by construction. The utility of the derived inference

Relationship	**Formula**
Key associated with Alice	$K_a \Rightarrow \textit{Alice}$
Bob has jurisdiction (controls or is believed) over statement φ	$\textit{Bob}$ controls φ
Alice and Bob together say φ	$(\textit{Alice} \,\&\, \textit{Bob})$ says φ
Alice quotes Bob as saying φ	$(\textit{Alice} \mid \textit{Bob})$ says φ
Bob is Alice's delegate on statement φ	$\textit{Bob}$ reps $\textit{Alice}$ on φ
Carol is authorized in Role on statement φ	$\textit{Carol}$ reps $\textit{Role}$ on φ
Carol acting in Role makes statement φ	$(\textit{Carol} \mid \textit{Role})$ says φ

Fig. 6: Relationships and their description in the calculus

1. $\textit{CA}$ controls $(K_a \Rightarrow \textit{Alice})$	Trust in CA's jurisdiction over keys
2. $K_{ca} \Rightarrow \textit{CA}$	Trust assumption associating K_{ca} with CA
3. K_{ca} says $(K_a \Rightarrow \textit{Alice})$	Public key certificate
4. $\textit{CA}$ says $(K_a \Rightarrow \textit{Alice})$	2, 3 Derived Speaks For
5. $K_a \Rightarrow \textit{Alice}$	1, 4 Controls

Fig. 7: Proof for associating a public key with a principal

1. $\textit{BFC}$ controls $\langle go \rangle$	Trust in BFC's authority
2. $\textit{Alice}$ reps $\textit{BFC}$ on $\langle go \rangle$	Trust assumption that Alice is BFC
3. $\textit{CA}$ controls $(K_a \Rightarrow \textit{Alice})$	Trust in CA's jurisdiction over keys
4. $K_{ca} \Rightarrow \textit{CA}$	Trust assumption associating K_{ca} with CA
5. K_{ca} says $(K_a \Rightarrow \textit{Alice})$	Public key certificate
6. $(K_a \mid \textit{BFC})$ says $\langle go \rangle$	Alice's order as BFC encrypted using K_a
7. $K_a \Rightarrow \textit{Alice}$	3,4,5 DR1
8. $\textit{BFC} \Rightarrow \textit{BFC}$	Idempotency of $\Rightarrow$
9. $K_a \mid \textit{BFC} \Rightarrow \textit{Alice} \mid \textit{BFC}$	7, 8 Monotonicity of $\mid$
10. $\textit{Alice} \mid \textit{BFC}$ says $\langle go \rangle$	9, 6 Derived Speaks For
11. $\langle go \rangle$	1, 2, 10 Reps

Fig. 8: Proof of legitimacy of a mission order

$$\textit{DR1}\quad \frac{\begin{array}{c}\textit{CA} \text{ controls } (K_a \Rightarrow \textit{Alice}) \quad K_{ca} \Rightarrow \textit{CA} \\ K_{ca} \text{ says } (K_a \Rightarrow \textit{Alice})\end{array}}{K_a \Rightarrow \textit{Alice}}$$

$$\textit{DR2}\quad \frac{\begin{array}{c}\textit{BFC} \text{ controls } \langle go \rangle \qquad \textit{Alice} \text{ reps } \textit{BFC} \text{ on } \langle go \rangle \\ \textit{CA} \text{ controls } (K_a \Rightarrow \textit{Alice}) \qquad K_{ca} \Rightarrow \textit{CA} \\ K_{ca} \text{ says } (K_a \Rightarrow \textit{Alice}) \qquad (K_a \mid \textit{BFC}) \text{ says } \langle go \rangle\end{array}}{\langle go \rangle}$$

Fig. 9: Two derived inference rules

rules is twofold. First, they support reuse: once derived, the inference rule can be used in subsequent proofs (for example, DR1 is used in the mission-order proof of Figure 8). Second, they serve as operational checklists that can be used to justify actions when given a request/order within a specific context of trust assumptions, certificates, and jurisdictions of authorities.

C. Power Tools

No matter how simple a calculation might be, the possibility of human error is ever present. The utility of theorem provers—such as the Cambridge University Higher Order Logic (HOL-4) theorem prover [7]—is that they can be used to check proofs, serving as an antidote to self-delusion. Throughout the Cyber Engineering Semester, we viewed the use of HOL as a means for assurance, not as an end in itself.

Our experience was that a two-week introduction to functional programming in Haskell was sufficient for the CES students to begin working with HOL. With just this modest Haskell background, they could view the theorem prover as a collection of functional programs that return objects of type *theorem*. By mid-semester, they were able to use the HOL implementation of the access-control logic/command-and-control calculus. As an example, Figure 10 shows a verbatim interactive HOL session for the the proof of *DR1* that appeared in Figure 7. User inputs are on lines starting with the user prompt "–"; HOL's responses are on lines starting with ">".

Although the students initially found HOL challenging, they valued the assurance and deeper understanding they achieved by using HOL: whereas they had doubts about the correctness of their unchecked hand proofs, they were very confident once their proofs were checked in HOL. In addition to proving properties in the access-control logic using pre-defined inference rules, they were able to devise their own sound custom inference rules.

To be explicit, the CES students were *users* of the access-control logic implementation and calculus, not experts in HOL. We had embedded the logic's syntax and semantics, core inference rules, and the proofs of the underlying theorems as conservative extensions to HOL prior to the start of the semester. However, the students could wield HOL well enough in this specific domain to get useful results that they recognized to be useful.

IV. Results

In the previous section, we described the content of the Engineering Assurance Lab and the Cyber Engineering Seminar. We now turn our attention to what our students were able to do by the end of the semester.

A. Problems

The Cyber Engineering Seminar in combination with the AFRL Internship had three projects that all required formal analysis using the access-control logic to describe architectures, protocols, operations, and identify flaws. The three projects were (1) to analyze the Sony Play Station Network and its flaws; (2) to analyze a secure email protocol (RFC 1421); and (3) to analyze and correct two flawed protocols for secure communication and control in the cloud. Brief descriptions of the projects appear in Figure 11.

Project Descriptions
1) **Play Station Network:** Students had to formally model and validate the architecture of the Play Station Network (PSN). They then had to use their models to identify possible security flaws and recommend changes to eliminate the vulnerabilities. Vulnerabilities were generally found to be based on the trust assumptions.
2) **RFC 1421:** RFC 1421 describes a secure email protocol. Students were to read and translate the verbal description into a formal process diagram. This process diagram had to specify all the necessary steps required to successfully implement the protocol. Further, they modeled the resulting process using the access-control logic to identify the foundations for trust in the RFC.
3) **Internship project:** The students were to read two protocols previously created for secure communication and control in the cloud. These protocols are known to be incomplete and have flaws. Students used the access-control logic to model the system and identify the flaws from a formal design perspective.

Fig. 11: Cyber Engineering Seminar and internship projects

Assignments in the EAL were aimed at supporting the analysis required of students in the Cyber Engineering Seminar. Whereas the assignments in the first half of the EAL were dedicated to functional programming and proof techniques, the second half of the EAL was aimed at using the command-and-control calculus in HOL to do analysis of larger problems. This was one way we helped students "connect the dots" in terms of specification, design, and verification.

We describe one problem that was designed to help students execute the virtuous cycle of specification, design, and verification. The problem focuses on the command-and-control concept of operations (CONOPS) for a weapon that requires dual authorization for launch. The problem scenario involves coalition operations—Blue and Gold Forces—each with a commander, operator, and certificate authority (CA). A Joint Forces Certificate Authority certified each of the Blue and Gold Force CAs. The CA hierarchy and flow of command and control is shown in Figure 12. Each commander has authority to issue a go/nogo command, upon which the corresponding operators issued launch/abort commands to the weapon.

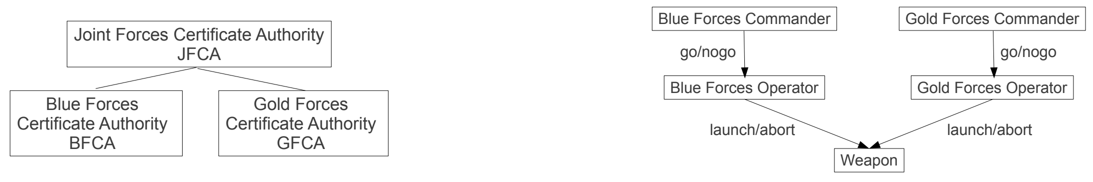
Figure 13 gives a high-level view of the CONOPS. The launch CONOPS shows the weapon being launched when both Blue and Gold Force Operators say *launch*. The abort CONOPS indicates that either Blue or Gold Force Operators can abort the weapon.

Given the high-level CONOPS as a start, the problem statement specified the following aspect:

1) Certificate authorities, keys, and means for authentication (see Figure 14)
2) Mission roles and the scope of authority (jurisdiction) for each role (see Figure 15)

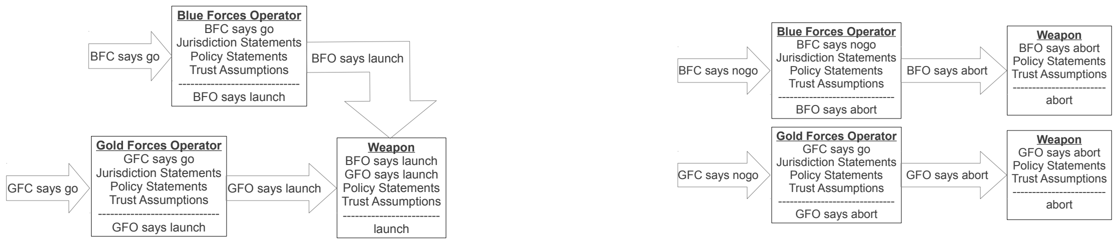
```
- val a1 = ACL_ASSUM ``(Name CA) controls ((Name Ka) speaks_for (Name Alice))``;
<<HOL message: inventing new type variable names: 'a, 'b, 'c>>
> val a1 =  [.] |- (M,Oi,Os) sat Name CA controls Name Ka speaks_for Name Alice
     : thm
- val a2 = ACL_ASSUM ``(Name Kca) speaks_for (Name CA)``;
<<HOL message: inventing new type variable names: 'a, 'b, 'c>>
> val a2 =  [.] |- (M,Oi,Os) sat Name Kca speaks_for Name CA : thm
- val a3 = ACL_ASSUM ``(Name Kca) says ((Name Ka) speaks_for (Name Alice))``;
<<HOL message: inventing new type variable names: 'a, 'b, 'c>>
> val a3 =  [.] |- (M,Oi,Os) sat Name Kca says Name Ka speaks_for Name Alice :
  thm
- val th1 = SPEAKS_FOR a2 a3;
> val th1 =  [..] |- (M,Oi,Os) sat Name CA says Name Ka speaks_for Name Alice :
  thm
- val th2 = CONTROLS a1 th1;
> val th2 =  [...] |- (M,Oi,Os) sat Name Ka speaks_for Name Alice : thm
```

Fig. 10: Sample proof in HOL of the derived inference rule DR1



(a) Certificate Authority Hierarchy (b) Flow of Command and Control

Fig. 12: Certificate-authority hierarchy and flow of command and control



(a) Launch CONOPS (b) Abort CONOPS

Fig. 13: High-level view of launch and abort CONOPS

3) Weapons launch and abort policy (see Figure 16)

4) Role assignments and authorizations (see Figure17)

Students defined and verified weapons launch and abort CONOPS using the access-control logic in HOL. They proved the validity of their authentication CONOPS based on role authorization and key assignments in HOL. Specifically, they had to devise the jurisdiction statements, policy statements, and trust assumptions; express them using the HOL implementation; and prove the validity of inference rules corresponding to weapons launch or abort in HOL. For this problem, students refined, described, and verified their CONOPS at three levels of abstraction: (1) the highest level with only roles, (2) a middle level with staff assigned to roles, and (3) the lowest level with cryptographic keys assigned to mission staff.

The final problem assigned to EAL students required them to do one more refinement: define the semantics of specific message and certificate formats using the access-control logic in HOL. This enabled them to describe and verify the behavior of their designs using concrete message and certificate formats.

Certificate Authority	Associated Key	How Authenticated
JFCA	K_{JFCA}	Pre-distributed to all mission principals
Blue Forces CA	K_{BFCA}	Authenticated by JFCA
Gold Forces CA	K_{GFCA}	Authenticated by JFCA

Fig. 14: Certificate authorities, keys, and authentication means

Role	Controls	How Authenticated
Blue Forces Commander	`go/nogo`	Pre-distributed to all Blue Forces mission principals
Gold Forces Commander	`go/nogo`	Pre-distributed to all Gold Forces mission principals
Blue Forces Operator	`launch/abort`	Blue Forces Commander
Gold Forces Operator	`launch/abort`	Gold Forces Commander

Fig. 15: Mission roles and jurisdiction

B. Proofs

The students' HOL proofs used the specialized inference rules we created for them as part of our HOL implementation of the access-control logic. We supplied 36 HOL inference rules that corresponded to the inference rules in the access-control logic textbook [4]. All of the proofs done for the more complex

Action	Command and Control Requirements
Weapons Launch	*Launch* ordered by both Blue and Gold Operators
Weapons Abort	*Abort* ordered by either Blue or Gold Operator

Fig. 16: Weapons launch and abort policy

Person	How Authenticated	Formal Description of Delegation of Authority
Alice as BFC	pre-distributed prior to mission	*Alice* reps *BFC* on φ
		Alice reps *BFC* on (*Carol* reps *BFO* on φ)
Bob as GFC	pre-distributed prior to mission	*Bob* reps *GFC* on φ
		Bob reps *GFC* on (*Dan* reps *GFO* on φ)
Carol as BFO	By Alice as BFC	*Carol* reps *BFO* on φ
Dan as GFO	By Bob as GFC	*Dan* reps *GFO* on φ

Fig. 17: Role assignments and authorizations

problems resembled the example shown in Figure 10. The proof support we supplied was intended to mimic what they would do in their pencil-and-paper proofs.

One group did a complete analysis of the secure cloud protocols for their internship using the HOL implementation of the access-control logic. They did this on their own initiative without any additional help from the faculty teaching the Engineering Assurance Laboratory.

C. *Reports*

Reports were an important deliverable for all projects: no one believes an analysis unless it is presented in a compelling fashion.

We required students to use LaTeX and Beamer for all reports and presentations in both the Cyber Engineering Seminar and Engineering Assurance Laboratory. One primary motivation for this requirement was to make use of HOL's automatic generation of LaTeX macros that typeset all formulas of a theory: datatypes, definitions, and theorems. These macros mean that HOL users do not have to manually enter the formulas in their documentation, thereby eliminating a major source of error. The benefit to readers and users of engineering reports is the high degree of assurance and confidence that what is documented is accurate and free from error.

Given the large number of formulas to be documented and typeset, we found that the benefits of using LaTeX and Beamer far outweighed the costs associated with teaching the students these tools. We provided students with style files and sample report/presentation templates. The reports and presentations were professional in appearance. The accuracy of the HOL documentation was assured due to the use of HOL's LaTeX macros for pretty printing HOL theories.

V. Plans: Lessons Learned and Next Steps

As an educational experiment, the goal of the 2011 Cyber Engineering Semester pilot program was to see if we could get undergraduates to *rigorously assure* the security and integrity of computer systems they devised at the level of hardware, architecture, operating systems, networks, protocols, and applications. Simply put, would they be able to rigorously comprehend, analyze, and synthesize the concepts put forth in Saltzer and Schroeder's classic paper, *The Protection of Information in Computer Systems* [8]?

As instructors, we had three major questions: (1) Were we asking too much? (2) Would the design of our five courses in fact reinforce security and integrity concepts so that students would be able to "connect the dots"? (3) Would students actually be able to do formal proofs of correctness using tools (e.g., theorem provers) generally thought to be beyond the grasp of undergraduates? The answers in short are: (1) almost, (2) mostly, and (3) yes.

A. *Lessons Learned*

We learned multiple lessons that will inform our next steps.

1) The CES courses and internship were very intense, due in large part to the time demands of class meetings, internship responsibilities, and homework. Because our students had excellent skills (all had GPAs of 3.3 and above), they coped. However, the small size of the pilot program also allowed us to make some minor mid-semester adjustments to help alleviate the time pressure. Our challenge and goal is to convey the same capabilities next time with less intensity.
2) All five courses utilized the access-control logic or its concepts. The benefit was a common way of thinking about security and integrity across the five courses. Mutually reinforcing assignments, when we were able to synchronize them, worked.
3) Introducing HOL theorem proving worked because students had enough functional-programming expertise to comprehend, analyze, and synthesize HOL inference rules as functional programs. Restricting students' attention primarily to the 36 access-control logic inference rules enabled the students to do meaningful HOL proofs. The value to the students was the assurance they achieved when they proved their CONOPS were valid.

B. *Next Steps*

Based on our experience, we have a much better idea of what undergraduates are able to accomplish and how we might simplify our program for 2012. Specifically, we can accurately plan the timing of our assignments to smooth out the workload of students. We plan on having hardware, OS, and architecture projects more closely synchronized to avoid duplication and promote greater reinforcement of common ideas. In terms of formal verification and functional programming, we now know how much detail is needed in our explanations and exercises. This will enable us to move more quickly with greater coverage of concepts.

All the partners were pleased with the outcome of the pilot. We are planning on testing our latest ideas in the Summer 2012 Information Assurance Internship in preparation for the Fall 2012 CES. Finally, we are planning a four-year cyber engineering program with an eye toward making the assurance of security and integrity in cyber systems a routine part of computer engineering and computer science at the undergraduate level.

VI. Conclusions

"*. . . officers can never act with confidence until they are masters in their profession . . .*" – Col. Harry Knox, 27 September 1776, on establishing the need for artillery schools in the US

Colonel Knox's observations in 1776 apply today: *rigorous* education is essential, because there is no substitute for knowing what you are doing. Just as artillery officers need to know the mathematics of ballistic trajectories, doctrine, and solutions to operational and tactical problems, cyberspace leaders need to know the underlying mathematics of cyberspace, be able to solve operational and tactical problems, and develop effective doctrine.

Our goal—from the start of the 2003 Advanced Course in Engineering Cyber Security Boot Camp, through the Cyber Engineering Semester, and onwards to a four-year Cyber Engineering Program—is to meld systems engineering thinking with leadership thinking. Cyberspace leaders who can "do the math" as systems engineers and military leaders are capable of reshaping cyberspace to their advantage.

Our thinking on this subject has evolved and expanded beyond what we initially thought possible. We have needed the ten years to get as far as we have, and our efforts have benefitted greatly from the ability to try out our ideas on students from a wide variety of institutions. When we began in 2003, we thought that the notion of having rising juniors use a command-and-control calculus with a Kripke semantics was highly questionable. In 2005, we discovered that eight hours of instruction could equip students to use the calculus correctly; however, such limited instruction time was insufficient for students to carry on by themselves without structured supervision. By the 2011 IA Internships, the instruction had grown to ten weeks and included functional programming and just a hint of theorem proving. The Cyber Engineer Semester provided sufficient evidence that mathematically rigorous assurance is feasible in a way that is relevant beyond one course.

Two professors who can do a proof is not a critical capability. In contrast, *two thousand* engineers and officers capable of mathematical analysis and leadership in support of designing, verifying, procuring, and operating systems *is* a critical capability. This vision is shared by an academic, industry, and governmental partnership with a ten-year history of joint operations. By working together to realize this vision, we have gained a level of trust and conceptual unity that enables us to rapidly design and deploy research and educational programs, integrated with internships and focused on real-world problems.

We are working hard to meet the goal of two thousand engineers and officers with the capabilities described here. We hope to expand this partnership to other universities, corporations, and government agencies so that we together can meet the critical national need for capable leadership in cyberspace.

References

[1] Dan Carnevale, "Basic training for anti-hackers: An intensive summer program drills students on cybersecurity skills," *The Chronicle of Higher Education*, September 23 2005.

[2] Kamal Jabbour and Susan Older, "The advanced course in engineering on cyber security: A learning community for developing cyber-security leaders," in *Proceedings of the Sixth Workshop on Education in Computer Security*, July 2004.

[3] American Society for Engineering Education, *The General Electric Advanced Course in Engineering*, Colorado State University, Ft. Collins, CO 80521, June 16–19 1975.

[4] Shiu-Kai Chin and Susan Older, *Access Control, Security, and Trust: A Logical Approach*, CRC Press, 2011.

[5] Butler Lampson, Martin Abadi, Michael Burrows, and Edward Wobber, "Authentication in distributed systems: Theory and practice," *ACM Transactions on Computer Systems*, vol. 10, no. 4, pp. 265–310, November 1992.

[6] Martin Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin, "A calculus for access control in distributed systems," *ACM Transactions on Programming Languages and Systems*, vol. 15, no. 4, pp. 706–734, September 1993.

[7] M.J.C. Gordon and T.F. Melham, *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*, Cambridge University Press, New York, 1993.

[8] Jerome Saltzer and Michael Schroeder, "The protection of information in computer systems," *Proceedings of IEEE*, 1975.