Policy-Based Design and Verification for Mission Assurance*

Shiu-Kai Chin[1], Sarah Muccio[2], Susan Older[1], and Thomas N. J. Vestal[2]

[1] EECS Department, Syracuse University, Syracuse, New York 13244, USA
[2] Air Force Research Laboratory, Rome, New York 13441, USA

Abstract. Intelligent systems often operate in a blend of cyberspace and physical space. Cyberspace operations—planning, actions, and effects in realms where signals affect intelligent systems—often occur in milliseconds without human intervention. Decisions and actions in cyberspace can affect physical space, particularly in SCADA—supervisory control and data acquisition—systems. For critical military missions, intelligent and autonomous systems must adhere to commander intent and operate in ways that assure the integrity of mission operations. This paper shows how policy, expressed using an access-control logic, serves as a bridge between commanders and implementers. We describe an access-control logic based on a multi-agent propositional modal logic, show how policies are described, how access decisions are justified, and give examples of how concepts of operations are analyzed. Our experience is policy-based design and verification is within the reach of practicing engineers. A logical approach enables engineers to think precisely about the security and integrity of their systems and the missions they support.
Key words: policy, concept of operations, access control, logic

1 Introduction

Cyber space and physical space are ever more intertwined. Cyber-physical systems, i.e., systems with tight coordination between computational and physical resources, operate in these intertwined worlds. Automatic pilots in aircraft and smart weapons are examples of cyber-physical systems where the capability to complete Boyd's *observe-orient-decide-act* decision loop [1] in milliseconds without human intervention is essential.

For commanders, fulfilling the missions entrusted to them is of paramount importance. As autonomous cyber and cyber-physical systems have by their very nature little, if any, human supervision in their decision loops, mission assurance and mission integrity concerns require that the trustworthiness of these systems be rigorously established.

A practical concern is how commanders and implementers will communicate with each other. Commanders operate at the level of policy: what is permitted and under what circumstances. Implementers are concerned with mechanisms. Our observation is that commanders and implementers communicate through descriptions of policy and concepts of operation. Our key contribution is a methodology for describing policies and trust assumptions within the context of concepts of operations.

* Distribution Statement A—Approved for Public Release—Distribution Unlimited Document #88ABW-2010-0819, dated 24 February 2010

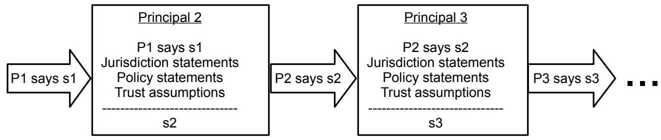

Fig. 1. Concept of Operations

The remainder of this paper is organized as follows. First, we informally describe the central elements of policy and concepts of operation that we wish to describe and justify rigorously. Second, we describe the syntax and semantics of our access-control logic. Third, we describe a hypothetical concept of operations, formalize its description, and provide a formal justification for its operations. Finally, we offer summary remarks and conclusions.

2 Elements of Policy and Concepts of Operation

Policies are principles, guides, contracts, agreements, or statements about decisions, actions, authority, delegation, credentials, or representation. Concepts of operation (CONOPS) describe a system from the user's perspective. CONOPS describe the goals, objectives, policies, responsibilities, jurisdictions of various authorities, and operational processes.

The elements of policy we are concerned with include:

- who or what has control over an action and under what circumstances,
- what are recognized tokens of authority,
- who are recognized delegates,
- what credentials are recognized,
- what authorities are recognized and on what are they trusted, and
- any trust assumptions used in making decisions or judgments.

We conceptualize CONOPS as a chain of statements or requests for action. These requests are granted or rejected based on the elements of policy listed above. This is illustrated in Figure 1. What Figure 1 shows is an abstract depiction of a CONOPS that has three or more principals or agents: *P1, P2,* and *P3*. Principals are entities such as subjects, objects, keys, tokens, processes, etc. Principals are anything or anybody that makes requests, is acted upon, or is used as a token representing a principal.

CONOPS begin with a statement or request *s1* by *P1*. In the syntax of the access-control logic we introduce next, this is the formula $P1 \text{ says } s1$. Principal *P2*, is envisioned to receive the statement $P1 \text{ says } s1$, and within the context of jurisdiction statements, policy statements, and trust assumptions, *P2* concludes $s2$ is justified. As a result of this justification, principal *P2* transmits a statement $P2 \text{ says } s2$ to principal *P3*, who then reacts within the context of its jurisdiction and policy statements, and trust assumptions. We repeat this for all principals and processes in the CONOPS.

Within the boxes labeled *Principal 2* and *Principal 3* are expressions

$$\frac{\begin{array}{c} P1 \text{ says } s1 \\ \textit{Jurisdiction statements} \\ \textit{Policy statements} \\ \textit{Trust assumptions} \end{array}}{s2} \quad \text{and} \quad \frac{\begin{array}{c} P2 \text{ says } s2 \\ \textit{Jurisdiction statements} \\ \textit{Policy statements} \\ \textit{Trust assumptions} \end{array}}{s3}.$$

What the above expressions intend to convey is that based on: (1) the statements or requests $s1$ and $s2$ made by principals $P1$ and $P2$, and (2) the statements of jurisdiction, policy, and trust assumptions under which principals $P2$ and $P3$ operate, $P2$ and $P3$ are logically justified (using the logic and calculus we describe next) to conclude $s2$ and $s3$. As we will see after formally describing the syntax and semantics of our logic, the two expressions above have the form of *derived* inference rules or *theorems* in our calculus. Each step of a CONOPS expressed in this fashion is a theorem justifying the behavior of a system.

One of the principal values of using the access-control logic is the evaluation of a CONOPS for logical consistency within the context of given policies, certifications, and trust assumptions. The process we outline here makes explicit underlying assumptions and potential vulnerabilities. This leads to a deeper understanding of the underpinnings of security and integrity for a system. This greater understanding and precision, when compared to informal descriptions, produces more informed design decisions and trade-offs.

In the following section, we define the syntax and semantics of the access-control logic and calculus.

3 An Access-Control Logic and Calculus

3.1 Syntax

Principal Expressions Let P and Q range over a collection of principal expressions. Let A range over a countable set of simple principal names. The abstract syntax of principal expressions is:

$$P ::= A \;/\; P\&Q \;/\; P \mid Q$$

The principal $P\&Q$ ("P in conjunction with Q") is an abstract principal making exactly those statements made by both P and Q; $P \mid Q$ ("P quoting Q") is an abstract principal corresponding to principal P quoting principal Q.

Access Control Statements The abstract syntax of statements (ranged over by φ) is defined as follows, where P and Q range over principal expressions and p ranges over a countable set of *propositional variables*:

$$\begin{aligned} \varphi ::= p \;/\; \neg\varphi \;/\; \varphi_1 \wedge \varphi_2 \;/\; \varphi_1 \vee \varphi_2 \;/\; \varphi_1 \supset \varphi_2 \;/\; \varphi_1 \equiv \varphi_2 \;/ \\ P \Rightarrow Q \;/\; P \text{ says } \varphi \;/\; P \text{ controls } \varphi \;/\; P \text{ reps } Q \text{ on } \varphi \end{aligned}$$

Informally, a formula $P \Rightarrow Q$ (pronounced "P speaks for Q") indicates that *every* statement made by P can also be viewed as a statement from Q. A formula P controls φ is syntactic sugar for the implication (P says φ) $\supset \varphi$: in effect, P is a trusted authority with respect to the statement φ. P reps Q on φ denotes that P is Q's delegate on φ; it is syntactic sugar for (P says (Q says φ)) $\supset Q$ says φ. Notice that the definition of P reps Q on φ is a special case of controls and in effect asserts that P is a trusted authority with respect to Q saying φ.

$$
\begin{aligned}
\mathcal{E}_{\mathcal{M}}[\![p]\!] &= I(p)\\
\mathcal{E}_{\mathcal{M}}[\![\neg\varphi]\!] &= W - \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]\\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \wedge \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!]\\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \vee \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cup \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!]\\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \supset \varphi_2]\!] &= (W - \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!]) \cup \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!]\\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \equiv \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1 \supset \varphi_2]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\varphi_2 \supset \varphi_1]\!]\\
\mathcal{E}_{\mathcal{M}}[\![P \Rightarrow Q]\!] &= \begin{cases} W, & \text{if } J(Q) \subseteq J(P)\\ \emptyset, & \text{otherwise}\end{cases}\\
\mathcal{E}_{\mathcal{M}}[\![P \textsf{ says } \varphi]\!] &= \{w | J(P)(w) \subseteq \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]\}\\
\mathcal{E}_{\mathcal{M}}[\![P \textsf{ controls } \varphi]\!] &= \mathcal{E}_{\mathcal{M}}[\![(P \textsf{ says } \varphi) \supset \varphi]\!]\\
\mathcal{E}_{\mathcal{M}}[\![P \ \textsf{ reps } Q \textsf{ on } \varphi]\!] &= \mathcal{E}_{\mathcal{M}}[\![P \mid Q \textsf{ says } \varphi \supset Q \textsf{ says } \varphi]\!]
\end{aligned}
$$

Fig. 2. Semantics

3.2 Semantics

Kripke structures define the semantics of formulas.

Definition 1. *A* Kripke structure $\mathcal{M}$ *is a three-tuple* $\langle W, I, J\rangle$*, where:*

- W *is a nonempty set, whose elements are called* worlds*.*
- I : $\textbf{\textit{PropVar}} \rightarrow \mathcal{P}(W)$ *is an* interpretation *function that maps each propositional variable* p *to a set of worlds.*
- J : $\textbf{\textit{PName}} \rightarrow \mathcal{P}(W \times W)$ *is a function that maps each principal name* A *to a relation on worlds (i.e., a subset of* $W \times W$*).*

We extend J to work over arbitrary *principal expressions* using set union and relational composition as follows:

$$
\begin{aligned}
J(P\&Q) &= J(P) \cup J(Q)\\
J(P \mid Q) &= J(P) \circ J(Q),
\end{aligned}
$$

where

$$
J(P) \circ J(Q) = \{(w_1, w_2) \mid \exists w'.(w_1, w') \in J(P) \text{ and } (w', w_2) \in J(Q)\}
$$

Definition 2. *Each Kripke structure* $\mathcal{M} = \langle W, I, J\rangle$ *gives rise to a function*

$$
\mathcal{E}_{\mathcal{M}}[\![-]\!] : \textbf{\textit{Form}} \rightarrow \mathcal{P}(W),
$$

where $\mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$ *is the set of worlds in which* φ *is considered true.* $\mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$ *is defined inductively on the structure of* φ*, as shown in Figure 2.*

Note that, in the definition of $\mathcal{E}_{\mathcal{M}}[\![P \textsf{ says } \varphi]\!]$*,* $J(P)(w)$ *is simply the image of world* w *under the relation* $J(P)$*.*

3.3 Inference Rules

In practice, relying on the Kripke semantics alone to reason about policies, CONOPS, and behavior is inconvenient. Instead, inference rules are used to manipulate formulas in the logic. All logical rules must be sound to maintain consistency.

$$\textit{Taut} \quad \frac{}{\varphi} \quad \text{if } \varphi \text{ is an instance of a prop-logic tautology}$$

$$\textit{Modus Ponens} \quad \frac{\varphi \qquad \varphi \supset \varphi'}{\varphi'} \qquad \textit{Says} \quad \frac{\varphi}{P \textsf{ says } \varphi}$$

$$\textit{MP Says} \quad \frac{}{(P \textsf{ says } (\varphi \supset \varphi')) \supset (P \textsf{ says } \varphi \supset P \textsf{ says } \varphi')}$$

$$\textit{Speaks For} \quad \frac{}{P \Rightarrow Q \supset (P \textsf{ says } \varphi \supset Q \textsf{ says } \varphi)}$$

$$\textit{Quoting} \quad \frac{}{P \mid Q \textsf{ says } \varphi \equiv P \textsf{ says } Q \textsf{ says } \varphi}$$

$$\&\textit{Says} \quad \frac{}{P\&Q \textsf{ says } \varphi \equiv P \textsf{ says } \varphi \wedge Q \textsf{ says } \varphi}$$

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

$$\textit{Associativity of} \mid \quad \frac{P \mid (Q \mid R) \textsf{ says } \varphi}{(P \mid Q) \mid R \textsf{ says } \varphi}$$

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

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

Fig. 3. Core Inference Rules

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

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

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

$$\textit{Rep Says} \quad \frac{P \textsf{ reps } Q \textsf{ on } \varphi \quad P \mid Q \textsf{ says } \varphi}{Q \textsf{ says } \varphi}$$

Fig. 4. Derived Rules Used in this Paper

Definition 3. *A rule of form* $\dfrac{H_1 \cdots H_n}{C}$ *is sound if, for all Kripke structures* $\mathcal{M} = \langle W, I, J \rangle$, *if* $\mathcal{E}_{\mathcal{M}}[\![H_i]\!] = W$ *for each* $i \in \{1, \ldots, n\}$, *then* $\mathcal{E}_{\mathcal{M}}[\![C]\!] = W$.

The rules in Figures 3 and 4 are all sound. As an additional check, the logic and rules have been implemented in the HOL-4 (Higher Order Logic) theorem prover as a conservative extension of the HOL logic [2].

3.4 Confidentiality and Integrity Policies

Confidentiality and integrity policies such as Bell-LaPadula [3] and Biba's Strict Integrity policy [4], depend on classifying, i.e., assigning a confidentiality or integrity level to information, subjects, and objects. It is straightforward to extend the access-control logic to include confidentiality, integrity, or availability levels as needed. In what follows, we show how the syntax and semantics of *integrity* levels are added to the core access-control logic. The same process is used for levels used for *confidentiality* and *availability*.

Syntax The first step is to introduce syntax for describing and comparing security levels. **IntLabel** is the collection of *simple integrity labels*, which are used as names for the integrity levels (e.g., HI and LO).

Often, we refer abstractly to a principal P's integrity level. We define the larger set **IntLevel** of *all* possible integrity-level expressions:

$$\textbf{IntLevel} ::= \textbf{IntLabel} \;/\; \mathsf{ilev}(\textbf{PName}).$$

A integrity-level expression is either a simple integrity label or an expression of the form $\mathsf{ilev}(A)$, where A is a simple principal name. Informally, $\mathsf{ilev}(A)$ refers to the integrity level of principal A.

Finally, we extend our definition of well-formed formulas to support comparisons of integrity levels:

$$\textbf{Form} ::= \textbf{IntLevel} \leq_i \textbf{IntLevel} \;/\; \textbf{IntLevel} =_i \textbf{IntLevel}$$

Informally, a formula such as $\textsc{lo} \leq_i \mathsf{ilev}(\mathit{Kate})$ states that Kate's integrity level is greater than or equal to the integrity level LO. Similarly, a formula such as $\mathsf{ilev}(\mathit{Barry}) =_i \mathsf{ilev}(\mathit{Joe})$ states that Barry and Joe have been assigned the same integrity level.

Semantics Providing formal and precise meanings for the newly added syntax requires us to first extend our Kripke structures with additional components that describe integrity classification levels. Specifically, we introduce extended Kripke structures of the form

$$\mathcal{M} = \langle W, I, J, K, L, \preceq \rangle,$$

where:

- W, I, and J are as defined earlier.
- K is a non-empty set, which serves as the universe of *integrity levels*.
- $L : (\textbf{IntLabel} \cup \textbf{PName}) \to K$ is a function that maps each integrity label and each simple principal name to a integrity level. L is extended to work over arbitrary integrity-level expressions, as follows:

  $$L(\mathsf{ilev}(A)) = L(A),$$

  for every simple principal name A.
- $\preceq \subseteq K \times K$ is a partial order on K: that is, $\preceq$ is *reflexive* (for all $k \in K$, $k \preceq k$), *transitive* (for all $k_1, k_2, k_3 \in K$, if $k_1 \preceq k_2$ and $k_2 \preceq k_3$, then $k_1 \preceq k_3$), and *anti-symmetric* (for all $k_1, k_2 \in K$, if $k_1 \preceq k_2$ and $k_2 \preceq k_1$, then $k_1 = k_2$).

Using these extended Kripke structures, we extend the semantics for our new well-formed expressions as follows:

$$\mathcal{E}_{\mathcal{M}}[\![\ell_1 \leq_i \ell_2]\!] = \begin{cases} W, & \text{if } L(\ell_1) \preceq L(\ell_2) \\ \emptyset, & \text{otherwise} \end{cases}$$

$$\mathcal{E}_{\mathcal{M}}[\![\ell_1 =_i \ell_2]\!] = \mathcal{E}_{\mathcal{M}}[\![\ell_1 \leq_i \ell_2]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\ell_2 \leq_i \ell_1]\!].$$

As these definitions suggest, the expression $\ell_1 =_i \ell_2$ is simply syntactic sugar for $(\ell_1 \leq_i \ell_2) \wedge (\ell_2 \leq_i \ell_1)$.

$$\ell_1 =_i \ell_2 \stackrel{\text{def}}{=} (\ell_1 \leq_i \ell_2) \wedge (\ell_2 \leq_i \ell_1)$$

$$\textit{Reflexivity of } \leq_i \quad \frac{}{\ell \leq_i \ell}$$

$$\textit{Transitivity of } \leq_i \quad \frac{\ell_1 \leq_i \ell_2 \qquad \ell_2 \leq_i \ell_3}{\ell_1 \leq_i \ell_3}$$

$$sl \leq_i \quad \frac{\mathsf{ilev}(P) =_i \ell_1 \quad \mathsf{ilev}(Q) =_i \ell_i \quad \ell_1 \leq_i \ell_2}{\mathsf{ilev}(P) \leq_i \mathsf{ilev}(Q)}$$

Fig. 5. Inference rules for relating integrity levels

Logical Rules Based on the extended Kripke semantics we introduce logical rules that support the use of integrity levels to reason about access requests. Specifically, the definition, reflexivity, and transitivity rules in Figure 5 reflect that $\leq_i$ is a partial order. The fourth rule is derived and convenient to have.

4 Expressing Policy Elements in the Logic

With the definition of the syntax and semantics of access-control logic, we provide an introduction to expressing key elements of policy.

Statements and requests Statements and requests are made by principals. Requests are logical statements. For example, if Alice wants to read file *foo*, we represent Alice's request as $Alice$ says $\langle read, foo\rangle$. We interpret $\langle read, foo\rangle$ as "it would be advisable to read file *foo*."

Credentials or certificates are statements, usually signed with a cryptographic key. For example, assume we believe public key K_{CA} is the key used by certificate authority CA. With this belief, we would interpret a statement made by K_{CA} to come from CA. In particular, if K_{CA} says $(K_{Alice} \Rightarrow Alice)$, we would interpret this public key certificate signed by K_{CA} as having come from CA.

Jurisdiction Jurisdiction statements identify who or what has authority, specific privileges, powers, or rights. In the logic, jurisdiction statements usually are controls statements. For example, if Alice has the right to read file *foo*, we say $Alice$ controls $\langle read, foo\rangle$. If Alice has read jurisdiction on *foo* and Alice requests to read *foo*, then the *Controls* inference rule in Figure 4 allows us to infer $\langle read, foo\rangle$ is a sound decision, i.e.,

$$\frac{Alice \ \mathsf{controls}\ \langle read, foo\rangle \qquad Alice\ \mathsf{says}\ \langle read, foo\rangle}{\langle read, foo\rangle .}$$

Controls statements are also statements of trust. Suppose CA is recognized as the trusted authority on public-key certificates. If CA says $(K_{Alice} \Rightarrow Alice)$ then we believe that K_{Alice} is Alice's public key. An important consideration is that trust is not all or nothing in our logic. A principal may be trusted on some things but not others. For example, we may trust CA on matters related to Alice's key, but we may not trust CA on saying whether Alice has write permission on file *foo*. Essentially, the scope of trust of a principal is limited to the specific statements over which a principal has control.

Proxies and delegates Often, principals who are the sources of requests or statements, do not in fact make the statements or requests themselves to the guards protecting a resource. Instead, something or somebody makes the request on their behalf. For example, it is quite common for cryptographic keys to be used as proxies, or stand-ins, for principals. In the case of certificate authority CA, we would say $K_{CA} \Rightarrow CA$. If we get a certificate signed using K_{CA}, then we would attribute the information in that certificate to CA. For example, using the *Derived Speaks For* rule in Figure 4 we can conclude that certificate authority CA vouches for K_{Alice} being Alice's public key:

$$\frac{K_{CA} \Rightarrow CA \qquad K_{CA} \text{ says } (K_{Alice} \Rightarrow Alice)}{CA \text{ says } (K_{Alice} \Rightarrow Alice).}$$

In situations where delegates are relaying orders or statements from their superiors, we typically use reps formulas. For example, say Alice is Bob's delegate on withdrawing funds from $account_1$ and depositing funds into $account_2$. If we recognize Alice as Bob's delegate, we would write:

$$Alice \;\text{ reps } Bob \text{ on } (\langle withdraw\ \$10^6, account_1\rangle \wedge \langle deposit\ \$10^6, account_2\rangle).$$

From the semantics of reps, if we recognize Alice as Bob's delegate, in effect we are saying that Alice is trusted on Bob stating that he wishes a million dollars to be withdrawn from $account_1$ and deposited into $account_2$. If Alice says Bob says withdraw a million dollars from $account_1$ and deposit it into $account_2$, we will conclude that Bob has made the request. Using the *Rep Says* rule in Figure 4 we can conclude:

$$\frac{\begin{array}{c}Alice \;\text{ reps } Bob \text{ on } (\langle withdraw\ \$10^6, account_1\rangle \wedge \langle deposit\ \$10^6, account_2\rangle)\\ Alice \mid Bob \text{ says } (\langle withdraw\ \$10^6, account_1\rangle \wedge \langle deposit\ \$10^6, account_2\rangle)\end{array}}{Bob \text{ says } (\langle withdraw\ \$10^6, account_1\rangle \wedge \langle deposit\ \$10^6, account_2\rangle).}$$

5 An Extended Example

In this section we describe a hypothetical example CONOPS for joint operations where Joint Terminal Air Controllers (JTACs) on the ground identify targets and request they be destroyed. Requests are relayed to a theater command authority (TCA) by controllers in Airborne Early Warning and Control (AEW&C) aircraft. If approved by commanders, AEW&C controllers direct aircraft to destroy the identified target. To avoid threats due to compromised communications and control, the CONOPS specifies the use of a *mission validation appliance* (MVA) to authenticate requests and orders. What follows is a more detailed informal description of the scenario followed by a formalization and analysis of the CONOPS.

5.1 Scenario Description

The sequence of requests and approvals is as follows:

1. At the squad level, Joint Terminal Air Controllers (JTACs) are authorized to request air strikes against enemy targets in real time.
2. Requests are relayed to theater command authorities (TCAs) by Airborne Early Warning and Control (AEW&C) controllers.

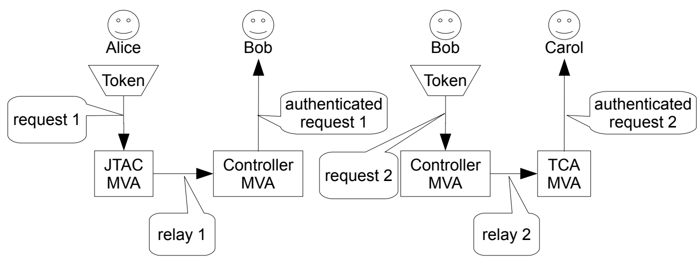

Fig. 6. Request Use Case

3. Requested air strikes are approved by TCAs. These commanders are geographically distant from the squad requesting an air strike.
4. Command and control is provided by AEW&C aircraft operating close to the squad requesting an air strike.

Threat Avoidance For mission security and integrity, JTACs, AEW&C controllers, pilots, and TCAs use a *mission validation appliance* (MVA) to request, transmit, authenticate, and authorize air strikes. MVAs are envisioned to be used as follows:

1. JTACs will use MVAs to transmit air strike requests to AEW&C controllers.
2. AEW&C controllers use MVAs to (a) authenticate JTACs, and (b) pass along JTAC requests to TCAs.
3. TCAs use MVAs to (a) authenticate JTACs and AEW&C controllers, and (b) send air strike authorizations to AEW&C controllers.
4. AEW&C controllers use MVAs to transmit air strike orders to pilots.

Security and Integrity Requirements The CONOPS for using MVAs must meet the following security and integrity requirements.

- All requests, commands, and approvals must be authenticated. *No voice communications will be used.* This includes at a minimum:
  - All personnel are to be authenticated into mission roles, i.e., joint terminal air controller (JTAC), airborne early warning and controller (AEW&C) controller, pilot, theater command authority (TCA) , and security officer (SO).
  - All communications, commands, and approvals are to be encrypted and signed for integrity.
- All aircraft pilots receive their directions from AEW&C controllers and can only act with the approval of the TCA.
- All keys, certificates, and delegations, i.e., the foundation for trust, must be protected from corruption during operations. Only personnel with proper integrity levels are allowed to establish or modify the foundation of trust.

Statement	Formal Representation
request 1	$(Token_{Alice} \mid JTAC)$ says $\langle strike, target\rangle$
relay 1	$(K_{JTAC\text{-}MVA} \mid JTAC)$ says $\langle strike, target\rangle$
authenticated request 1	$JTAC$ says $\langle strike, target\rangle$
request 2	$(Token_{Bob} \mid Controller)$ says $(JTAC$ says $\langle strike, target\rangle)$
relay 2	$(K_{Controller\text{-}MVA} \mid Controller)$ says $(JTAC$ says $\langle strike, target\rangle)$
authenticated request 2	$Controller$ says $(JTAC$ says $\langle strike, target\rangle)$

Table 1. Requests and Relayed Requests

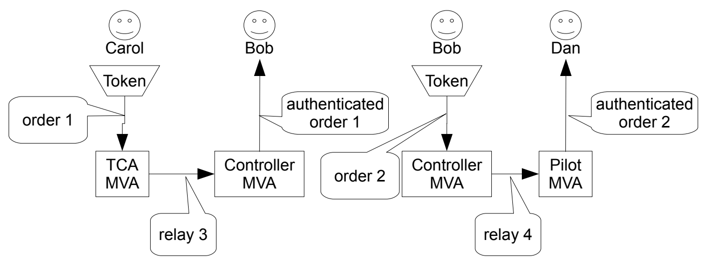

Fig. 7. Order Use Case

5.2 An Example CONOPS

MVA Use Cases We consider two use cases. The first use case shows how MVAs are used when an air strike is requested by a JTAC. The second use case shows how MVAs are used when a TCA orders an air strike. Figure 6 illustrates the flow of requests starting from Alice as JTAC, through Bob as Controller, resulting in an authenticated request to Carol as TCA. The process starts with Alice using her token $Token_{Alice}$ to authenticate herself and her request to the JTAC MVA. The JTAC MVA authenticates Alice and her role, and relays Alice's request using its key, $K_{JTAC\text{-}MVA}$ to the Controller MVA. The Controller MVA authenticates the JTAC MVA and presents the authenticated request to Bob.

Should Bob decide to pass on Alice's request, he uses his token to authenticate himself to the Controller MVA, which relays his request to the TCA MVA, which presents the authenticated request to Carol, a Theater Command Authority. Table 1 lists the formal representation of each request, relayed request, and authenticated request in Figure 6.

Figure 7 shows a similar flow of orders starting from Carol as TCA, through Bob as Controller, resulting in an authenticated order to Dan as Pilot. Carol authenticates herself to the TCA MCA using her token. Her orders are relayed to Bob. When Bob decides to pass on the order to Dan, he does so by authenticating himself to the Controller MVA, which relays to orders to Dan via the Pilot MVA. The formulation of each order and relayed order is shown in Table 2.

Deducing Policies, Certifications, Delegations, and Trust Assumptions Based on the use cases for air strike requests and air strike orders, we determine what

Statement	Formal Representation
order 1	$(Token_{Carol} \mid TCA)$ says $\langle strike, target \rangle$
relay 3	$(K_{TCA-MVA} \mid TCA)$ says $\langle strike, target \rangle$
authenticated order 1	TCA says $\langle strike, target \rangle$
order 2	$(Token_{Bob} \mid Controller)$ says $(TCA$ says $\langle strike, target \rangle)$
relay 4	$(K_{Controller-MVA} \mid Controller)$ says $(TCA$ says $\langle strike, target \rangle)$
authenticated order 2	$Controller$ says $(TCA$ says $\langle strike, target \rangle)$

Table 2. Orders and Relayed Orders

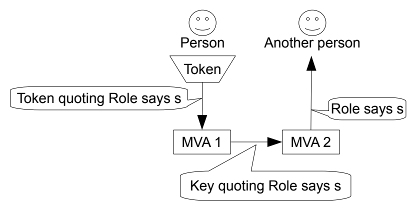

Fig. 8. General Pairing of MVAs

policies, certifications, delegations, and trust assumptions are required to justify each MVA action in the CONOPS. We look at each MVA's input and output, and based on the CONOPS, infer what policies, certifications, delegations, and trust assumptions are required. We look for repeated patterns of behavior that lead to repeated patterns of reasoning. Both use cases exhibit the same pattern of behavior as illustrated in Figure 8 and formulated in Table 3.

1. A person authenticates herself and claims a role using a token. Acting in a role, the person makes a statement (request or order). The first MVA, *MVA 1*, authenticates both the person and the role, and then relays the statement using its key to the second MVA, *MVA 2*.
2. *MVA 2* authenticates *MVA 1* and the role it is serving, then passes the statement up to the person using *MVA 2*.

Given the repeated pattern, we prove two *derived inference rules* (*MVA 1* and *MVA 2*) that justify the behavior of *MVA 1* and *MVA 2*.

$$\textit{MVA 1} \quad \frac{\begin{array}{c}(Token \mid Role) \text{ says } \varphi \\ K_{Auth} \text{ says } (Person \text{ reps } Role \text{ on } \varphi) \\ K_{Auth} \text{ says } (Token \Rightarrow Person) \\ Auth \text{ controls } (Person \text{ reps } Role \text{ on } \varphi) \\ Auth \text{ controls } (Token \Rightarrow Person) \\ K_{Auth} \Rightarrow Auth\end{array}}{K_{MVA_1} \mid Role \text{ says } \varphi}$$

$$\textit{MVA 2} \quad \frac{\begin{array}{c}(K_{MVA_1} \mid Role) \text{ says } \varphi \\ K_{Auth} \text{ says } (MVA_1 \text{ reps } Role \text{ on } \varphi) \\ K_{Auth} \text{ says } (K_{MVA_1} \Rightarrow MVA_1) \\ Auth \text{ controls } (MVA_1 \text{ reps } Role \text{ on } \varphi) \\ Auth \text{ controls } (K_{MVA_1} \Rightarrow MVA_1) \\ K_{Auth} \Rightarrow Auth\end{array}}{Role \text{ says } \varphi}$$

Statement	**Formal Representation**
statement	$(Token \mid Role)$ says φ
relayed statement	$(K_{MVA-1} \mid Role)$ says φ
authenticated statement	$Role$ says φ

Table 3. Statements and Relayed Statements

Item	**Formula**
Input	$(Token\ or\ Key \mid Role)$ says φ
Delegation Certificate	K_{Auth} says $(Person\ or\ Object$ reps $Role$ on $\varphi)$
Key Certificate	K_{Auth} says $(Token\ or\ Key \Rightarrow Person\ or\ Object)$
Jurisdiction	$Auth$ controls $(Person\ or\ Object$ reps $Role$ on $\varphi)$
Jurisdiction	$Auth$ controls $(Token\ or\ Key \Rightarrow Person\ or\ Object)$
Trust Assumption	$K_{Auth} \Rightarrow Auth$

Table 4. MVA Inputs, Outputs, Certificates, Jurisdiction, and Trust Assumptions

Both rules have the same components, as shown in Table 4. The components have the following functions:

1. *input*: a token or key quoting a role
2. *certificate*: a certificate authorizing a delegation
3. *certificate*: a public key certificate
4. *jurisdiction*: an assumption about an authority's jurisdiction to authorize a person or MVA to act in a role
5. *jurisdiction*: an assumption about an authority's jurisdiction over keys
6. *trust assumption*: knowledge of the trusted authority's key

Both rules have nearly identical proofs that are direct application of inference rules described in Section 3.3.

Using the inference rule *MVA 1*, we easily prove the following rule for the *TCA MVA* authenticating Carol and validating her order for an air strike, where *SO* is the *Security Officer* role, the *SO* has jurisdiction over roles and keys, and K_S is the key that speaks for the *SO*.

$$\textit{TCA-MVA}\quad \frac{\begin{array}{c} Token_{Carol} \mid TCA \text{ says } \langle strike, target\rangle \\ K_{SO} \text{ says } (Carol \text{ reps } TCA \text{ on } \langle strike, target\rangle) \\ K_{SO} \text{ says } Token_{Carol} \Rightarrow Carol \\ SO \text{ controls } Token_{Carol} \Rightarrow Carol \\ SO \text{ controls } (Carol \text{ reps } TCA \text{ on } \langle strike, target\rangle) \\ K_{SO} \Rightarrow SO \end{array}}{K_{\textit{TCA-MVA}} \mid TCA \text{ says } \langle strike, target\rangle}$$

Similar rules and proofs are written for each MVA. The above discussion on certificates installed properly in MVAs leads us to the final use case, namely the *trust establishment* use case.

5.3 Trust Establishment

Biba's Strict Integrity model [4] is the basis for maintaining integrity of the MVAs. As Strict Integrity is the dual of Bell and LaPadula's confidentiality

Role	Rights
SO (L_{Sec})	install, read
JTAC (L_{op})	read
Controller (L_{op})	read
TCA (L_{op})	read
Pilot (L_{op})	read

Table 5. Roles and Rights to Certificates

model [3], the short summary of Strict Integrity is, *no read down and no write up*. For subjects S and objects O, S may have discretionary read rights on O if O's integrity level meets or exceeds S's. For write access, S's integrity level must meet or exceed O's.

$$\mathsf{ilev}(S) \leq_i \mathsf{ilev}(O) \supset S \text{ controls } \langle read, O\rangle$$
$$\mathsf{ilev}(O) \leq_i \mathsf{ilev}(S) \supset S \text{ controls } \langle write, O\rangle.$$

There are two integrity levels: L_{op} and L_{Sec}, where $L_{op} \leq_i L_{Sec}$. All certificates have an integrity level L_{Sec}, i.e., $\mathsf{ilev}(cert) =_i L_{Sec}$. Table 5 show the integrity level and certificate access rights for each role. Strict integrity is satisfied as only the security officer SO (with the same integrity level L_{Sec} as certificates) can install or write certificates into MVAs. Every other role is at the L_{op} level and can only read certificates.

Installing K_{SO} Establishing the basis for trust in MVAs starts with the installation of the *Security Officer's* key, K_{SO}. This is assumed to be done by controlled physical access to each MVA that is deployed. Once the Security Officer's key is in place, the certificates that an MVA needs can be installed.

Certificate Installation Suppose Erica is acting as the Security Officer SO. The policy is that security officers can install certificates, if the SO has a high enough integrity level, and is given by

$$\mathsf{ilev}(cert) \leq_i \mathsf{ilev}(SO) \supset SO \text{ controls } \langle install, cert\rangle.$$

Erica's authorization to act in the Security Officer role to install certificates is given by

$$K_{so} \text{ says } Erica \text{ reps } SO \text{ on } \langle install, cert\rangle.$$

This authorization is accepted under the assumption that $K_{SO} \Rightarrow SO$ and that the SO has jurisdiction, which is given by

$$SO \text{ controls } Erica \text{ reps } SO \text{ on } \langle install, cert\rangle.$$

The proof for justifying Erica's capability to install certificates acting as a Security Officer, assuming her integrity level is L_{so} is a straightforward application of inference rules described in Section 3.3.

6 Related Work

The access-control logic we use is based on Abadi and Plotkin's work [5], with modifications described in [6]. Many other logical systems have been used to reason about access control. Some of them are summarized in [7].

Our contribution is the methodology and application of logic to describe policies, operations, and assumptions in CONOPS. Moreover, we have implemented this logic in the HOL-4 theorem prover, which provides both an independent verification of soundness as well as support for computer-assisted reasoning.

7 Conclusions

Our objective is the put usable mathematical methods into the hands of practicing engineers to help them reason about policies and concepts of operations. We have experimented with policy-based design and verification for five years in the US Air Force's Advanced Course in Engineering (ACE) Cybersecurity Bootcamps [8]. Our experience with a wide variety of students, practicing engineers, and Air Force officers suggests that using the access-control logic meets this objective.

References

1. Coram, R.: Boyd: The Fighter Pilot who Changed the Art of War. Back Bay Books/Little, Brown and Company (2002)
2. Gordon, M., Melham, T.: Introduction to HOL: A Theorem Proving Environment for Higher Order Logic. Cambridge University Press, New York (1993)
3. Bell, D.E., La Padula, L.J.: Secure computer systems: Mathematical foundations. Technical Report Technical Report MTR-2547, Vol. I, MITRE Corporation, Bedford, MA (March 1973)
4. Biba, K.: Integrity considerations for secure computer systems. Technical Report MTR-3153, MITRE Corporation, Bedford, MA (June 1975)
5. Abadi, M., Burrows, M., Lampson, B., Plotkin, G.: A Calculus for Access Control in Distributed Systems. ACM Transactions on Programming Languages and Systems **15**(4) (September 1993) 706–734
6. Chin, S.K., Older, S.: Reasoning about delegation and account access in retail payment systems. In: MMM-ACNS. (2007)
7. Abadi, M.: Logic in access control (tutorial notes). (2009) 145–165
8. Chin, S.K., Older, S.: A rigorous approach to teaching access control. In: Proceedings of the First Annual Conference on Education in Information Security, ACM (2006)