A Logical Representation of Common Rules for Controlling Access to Classified Information

Deepak Garg Frank Pfenning
Denis Serenyi† Brian Witten†

June 19, 2009
CMU-CS-09-139

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

†Author affiliation: Symantec Corporation

This work was partially supported by the Air Force Research Laboratory under grant number FA87500720028. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory, the U.S. Government, Symantec Corporation or Carnegie Mellon.

Keywords: Access control, policy, logic, classified information

Abstract

Official policies for controlling access to classified information in the U.S. are quite complex and often difficult to enforce. We present an encoding of a common core of these policies in an authorization logic, and describe their rigorous enforcement in PCFS, a file system implemented for such purposes.

1 Introduction

There is a significant mismatch in the complexity of policies for access to classified information in the U.S., and the sophistication of mechanisms currently available for their enforcement. Whereas policies often rely on high level concepts like delegation of rights, time-based expiration of credentials, and attributes of individuals and files, they are enforced by either physically separating networks carrying information classified at different levels, or through reference monitors based on access control lists. In practice, both these methods are problematic. Physically separated networks require individuals to *manually move data between machines* whenever they wish to work with information classified at different levels. On the other hand, access control lists are *difficult to administer* – capturing the high-level intents of policies in lists and keeping the lists up-to-date with changing credentials of both files and users requires a lot of manual effort and is a source of many errors. These problems suggest the need for a mechanism for representing policies at a high-level of abstraction and for enforcing them directly, without the need for physical separation or translation to access control lists, and minimal human intervention.

Based on these considerations the first two authors have designed and prototyped a file system, PCFS, that builds on proof-carrying authorization (PCA) [1], a promising, open-ended architecture for direct enforcement of high-level access control policies through reference monitors. In PCA, policy rules are represented as logical formulas at a high level of abstraction and enforced with formal proofs. Human intervention is needed only to create credentials that certify attributes of files and users. The technology has previously been applied to web services and distributed access control for physical devices [2, 3]. PCFS incorporates PCA in a file system. In addition, it includes a new, expressive logic for representing access policies. This logic, BL, supports not only standard policy constructs like delegation and decentralized administration, but also time-dependent credentials and state-dependent rules. Relying on a combination of PCA and cryptographic capabilities, PCFS enforces policies expressed in this rich logic in a correct and efficient manner.

This paper is a case study for the use of PCFS and BL. Its purpose is to show that common rules for disseminating classified information in the U.S. can be represented in BL and enforced with PCFS. BL and PCFS are presented only to the extent necessitated by this goal; their details may be found in a companion paper [8]. We believe that this paper also constitutes the first systematic exploration of formal foundations for access to classified information. Policies similar to those presented in this paper may be used in PCFS to protect proprietary information in other organizations as well.

The rest of this paper is organized as follows. In Section 2 we summarize PCFS and BL. Section 3 provides an overview of classified information, including the life cycle of a sensitive file and the high level policy rules for access to sensitive files. Section 4 describes the process of file classification, relevant properties of a classified file, and formal rules for establishing these properties. Section 5 presents rules for giving security clearances to individuals; these clearances are necessary to read classified files. Section 6 explains how properties of a classified file and security clearances of individuals interact to allow access. The appendix lists all logical predicates used in this paper, together with their intuitive meanings and the sections in which they are defined.

Methods and sources. Most of the policies formalized in this paper are derived from interviews of intelligence personnel conducted by Witten and Serenyi. Summarized results of these interviews were provided to the other two authors (in the form of detailed internal reports), who then formalized the policies in BL. Some parts of policies are based on Executive Orders of the White House [12, 13], or Director of Central Intelligence Directives (DCIDs) [10, 11]. Due to this mixed source of information, we do not explicitly cite our sources again in the paper. None of the information on which this paper relies is classified.

Limitations. This paper is not intended to be an authoritative reference on policies for controlling access to classified information, or of the actual practices followed for their enforcement. Instead, it presents a formalization of a selected and somewhat simplified set of these policies in BL. The primary intention of the paper is to show that BL is expressive enough to encode a large, representative part of the policies (which can then be enforced directly in PCFS), and to present, by example, general techniques for encoding other similar policies.

2 Summary of PCFS and BL

This section briefly discusses the syntax of the logic BL which is used for formalizing policies in this paper, and its enforcement in our file system PCFS (PCFS stands for Proof-Carrying File System). Details of both BL and PCFS may be found in prior work [6, 8].

PCFS allows direct enforcement of high-level, complex access policies expressed in the logic BL. The advantage of expressing policies in a logic (as opposed to, say, in access control lists) is that common policy idioms like conditional authorizations, delegation, attributes, groups, and time-based expiration can be represented directly, and rigorously interpreted via the logic's inference rules, without the need for human intervention. Building and improving upon prior work on authorization logics [4, 7, 9], BL supports three constructs in addition to the usual connectives of intuitionistic first-order logic: (a) The modality $k \mathsf{\ says\ } s$, which means that principal k states or believes that the policy or fact s holds (other principals may or may not believe s), (b) The modality s @ $[u_1, u_2]$, which means that credential or policy s holds from time u_1 to time u_2, but possibly not outside of the interval $[u_1, u_2]$, and (c) State predicates written in **boldface** which capture the state of the file system, such as parts of the metadata of a file. The syntax of BL is summarized below. s denotes a formula, p denotes an atomic formula, i denotes a state predicate, k denotes a principal, and u denotes a time point (externally represented as a clock time yyyy:mm:dd:hh:mm:ss, and internally represented as an integer measuring seconds from a fixed clock time).

$$s \ ::= \ p \mid i \mid s_1 \supset s_2 \mid s_1 \wedge s_2 \mid s_1 \vee s_2 \mid \top \mid \bot \mid \forall x{:}\sigma.s \mid \exists x{:}\sigma.s \mid k \mathsf{\ says\ } s \mid s \ @ \ [u_1, u_2]$$

Enforcement. Enforcement of policies in PCFS is based on proofs, an idea borrowed from prior work on proof-carrying authorization [1, 2]. Policy rules and credentials may be created by different individuals – a principal k may assert the fact s (representing a credential or a policy rule) by writing s in a digital certificate using a stipulated concrete syntax for BL, adding begin and end times u_1 and u_2 respectively to the certificate, and signing the certificate with her private key. This fact then gets reflected in BL as the formula $(k \textsf{ says } s) \ @\ [u_1, u_2]$. The collection of all such prevalent certificates is called the policy, denoted Γ. Principal K is allowed permission P ($\textsf{read}$, $\textsf{write}$, etc.) on file F at time u if and only if the policy Γ at time u *entails* in the proof system of the logic the formula $(\textsf{admin says may}(K, F, P)) \ @\ [u, u]$. Here, $\textsf{may}$ is a fixed predicate, and $\textsf{admin}$ is a distinguished principal who is assumed to have ultimate authority on giving access to files.[1] The inference rules of BL ensure that statements of distinct principals do not interfere unless explicitly stipulated by rules, and that time intervals in @ connectives are respected. The latter implies that proofs used for access automatically expire in PCFS, without need for explicit revocation or extra-logical checks on time bounds. A slight complication arises because in PCFS proofs are exchanged for capabilities ahead of access, but this detail is irrelevant to the work in this paper.

BL supports arithmetic constraints of the form $u_1 \leq u_2$, and two primitive time points $-\infty$ and $+\infty$ with the properties that $-\infty \leq u$ and $u \leq +\infty$ for every time point u. In addition, BL includes a strongest principal ℓ, with the property that $\ell \textsf{ says } s$ implies $k \textsf{ says } s$ for every k and s. ℓ may be used to state facts that are globally true in the system (see Section 5.2 for an example).

In addition to separating policies of different individuals through the $\textsf{says}$ connective, and allowing explicit time through the @ connective, BL also allows policies to depend on the state of the file system through state predicates. Natively, two state predicates are supported: $\mathbf{owner}(F, K)$ which holds whenever file F has owner K, and $\mathbf{has\_xattr}(F, A, V)$ which holds whenever file F has extended attribute A set to V. The latter allows representation of very general information about the state of a file, and may be useful in enforcing a wide variety of policies, including those in this paper (see Section 3.3 for an example). Support for other predicates can be added through a simple programming API provided by PCFS.

Some relevant consequences of BL's inference rules are summarized below (see [8] for details of the proof system). $\vdash s$ means that s can be proved without making any assumptions in BL's proof system. As a convention we assume that implication $\supset$ is right associative.

1. $\vdash (k \textsf{ says } (s_1 \supset s_2)) \supset (k \textsf{ says } s_1) \supset (k \textsf{ says } s_2)$

2. $\vdash (k \textsf{ says } s) \supset (k' \textsf{ says } k \textsf{ says } s)$

3. $\vdash (\ell \textsf{ says } s) \supset (k \textsf{ says } s)$

4. $(u_1 \leq u_1') \supset (u_2' \leq u_2) \supset (s \ @\ [u_1, u_2]) \supset (s \ @\ [u_1', u_2'])$

[1]Strictly speaking, the *judgment* $(\textsf{admin says may}(K, F, P)) \circ [u, u]$ must be established, but this judgment is internalized by and hence equivalent to the formula $(\textsf{admin says may}(K, F, P)) \ @\ [u, u]$. See [8] for details.

5. $\vdash ((s_1 \supset s_2) \ @ \ [u_1, u_2]) \supset (\forall u_1', u_2'. \ (u_1 \leq u_1') \supset (u_2' \leq u_2) \supset (s_1 \ @ \ [u_1', u_2']) \supset (s_2 \ @ \ [u_1', u_2']))$

The last property above means that if $(s_1 \supset s_2)$ holds throughout the interval $[u_1, u_2]$, then for any *subinterval* $[u_1', u_2']$ on which s_1 holds, s_2 also holds.

Notational conventions. As mentioned earlier, the general form of a policy rule or credential in BL is $(k \ \mathsf{says} \ s) \ @ \ [u_1, u_2]$. However, for all rules presented in this paper, $u_1 = -\infty$ and $u_2 = +\infty$, so we simply omit the suffix $@ \ [u_1, u_2]$. Further, we omit all sorts here and use logic programming notation to make the rules easy to read. s :- s' denotes $s' \supset s$, and s, s' denotes $s \wedge s'$. Conjunction binds tighter than implication, so s :- $s_1, \ldots, s_n$ denotes $(s_1 \wedge \ldots \wedge s_n) \supset s$. On the whole, the general form of rules presented in this paper is $k \ \mathsf{says} \ (s$:- $s_1, \ldots, s_n)$, which translates to $(k \ \mathsf{says} \ ((s_1 \wedge \ldots \wedge s_n) \supset s)) \ @ \ [-\infty, +\infty]$ in the syntax presented above. The formulas $s_1, \ldots, s_n$ are called the *conditions* of the rule. Any variables starting with uppercase letters (e.g., K, SCG, etc.) are assumed to be universally quantified *immediately inside* the annotation $k \ \mathsf{says} \ \cdot$.

We follow a descriptive naming convention for predicates. A predicate name has the form entity/attribute/..., where entity determines the entity whose attribute the predicate describes and attribute is a description of the property the predicate defines. "..." may be any other relevant qualifiers. Common among these is h which denotes a helper predicate that is used in the definition of the predicate without the h. Finally, we Curry arguments to predicates, writing, for example, may $K \ F \ P$ instead of may(K, F, P).

3 Sensitive Information Life Cycle

This section provides an overview of classification and declassification of information in the U.S. Unfortunately, some of the concepts and methods involved in classification are themselves classified and inaccessible to us. What follows is an abstracted and simplified description of some of the publicly available concepts.

The first salient point about classification is that depending on the structure of information, the amount of data classified together may vary: entire files, pages, or paragraphs may be marked for classification as a unit. In this paper, we assume that the unit of classification is a digital file, since it is easy to control access at that level via the file system.

A typical sensitive file created by an intelligence or defense agency goes through the life cycle depicted in Figure 1. There are 4 distinct states, which we discuss below. Transitions between these states are discussed in Section 3.2.

- **Default.** Every newly created file starts in a temporary state, which we call the default state. Only the individual creating the file has read and write access to a file in this state. A default file may subsequently either be designated a working paper, or it may be deleted.

- **Working paper.** A working paper is a file that will eventually be classified, but whose content has not been finalized. When in this state, read and write access to the

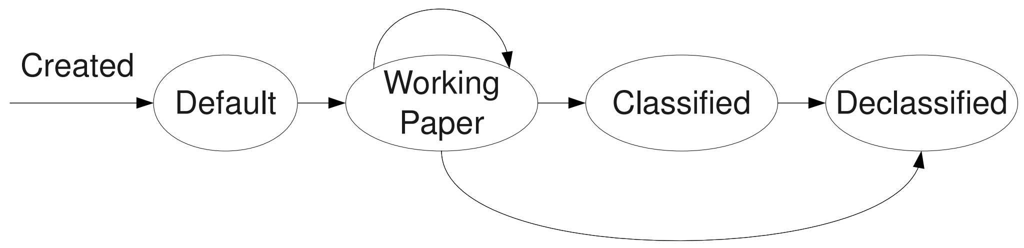

Figure 1: States of a sensitive file

file is at the discretion of the agency or group that is working on the file. A file stays in this state for at most 90 days, after which it must either be classified, reviewed again and placed in the same state, declassified, or deleted.

- **Classified.** After the content of a working paper is finalized, it is classified. Read access to a classified file is based on several properties of the classification (e.g., secrecy level, compartments, etc.) that are decided when the file is classified. These properties are discussed in Section 4. In addition, the agency that owns the file must authorize every read access to a classified file. Official guidelines do not specify who, if anyone, has write access to a classified file. Since changing the content of a classified file may require reclassification, it seems reasonable to assume that classified files cannot be written, and we make this assumption throughout this paper. Owing to concerns of accountability, we also assume that a classified file cannot be deleted.

- **Declassified.** A file may be released to the public (declassified) in two ways: (a) through an executive order, (b) through an automatic expiration of the classification at a stipulated point of time. In this paper we make the simplifying assumption that a declassified file may be read by anyone. In actual practice, a file may be declassified to specific groups of people, e.g., U.S. citizens. As for classified files, we assume that declassified files cannot be deleted.

3.1 Representation of File State in PCFS

To represent the state of a file, we use extended attributes that are natively supported in PCFS. The state predicate (**has_xattr** F A V) holds in BL if and only if the file F has the extended attribute named A set to the term V. V can be any term in BL. We use a specific extended attribute `status` with different values to record the state of a file. These values and their meanings are summarized below.

Value of extended attribute status on file F	Meaning
default	F is in default state
working T	F is a working paper, put into that state at time T
classified T T'	F is classified, effective from time T to time T'
declassified	F is declassified

When a file is created PCFS automatically sets its extended attribute status to default, and its owner to the principal who creates the file.[2] The time point T in the value working T represents the time at which the working paper state is effective. This is important because a working paper can be read and written only for 90 days after it enters the state. Similarly, the time point T' in classified T T' is necessary to determine when the classification will expire. If a classification lacks a fixed expiration, the BL constant $+\infty$ may be used for T'.

3.2 File State Transition

A central question regarding file states is *who* changes the states of a file. PCFS supports administrative roles for such purposes by requiring a special permission called govern for modifying any extended attribute of a file, or its owner.[3] An individual who has this permission on a file may use either the standard Linux system call `setxattr` or the command line program `setfattr` to change extended attributes of the file. For our proposed enforcement we assume that only a special principal sysadmin (intended to represent a system administrator) is allowed the govern permission on all files, as formalized by the following rule:

admin says (may sysadmin F govern).

No other rule in our formalization allows the govern permission on any file. Hence sysadmin alone may change the status of a file, and affect its state. Of course, state changes cannot be made ad hoc; sysadmin must perform these transitions only under certain conditions. The conditions necessary for each transition in Figure 1 are listed below, together with additional changes that must be made with the transition. Unfortunately, since PCFS only performs access control on files, these conditions cannot be enforced by PCFS. Instead, we must assume that sysadmin follows these guidelines accurately.

- Default to working paper: This transition may be applied at the discretion of the owner (creator) of the file. The status of the file must be set to working T where T is the time at which the transition is applied, and the owner of the file must be changed from the creator of the file to the agency or group that is working on the file (or their representative).

[2] In the actual implementation of PCFS, the attribute status is called newfile, and the value default is 1. Since the difference is merely cosmetic, we use the more meaningful names status and default here.

[3] In contrast, in a POSIX compliant system, the common write permission suffices for these purposes.

- Working paper to working paper: The purpose of this transition is to extend the 90 day working period of a file. It may be applied at the discretion of the file's owner, which will be the group or agency working on the file. The status of the file must be set to working T where T is the time at which the transition is applied.

- Working paper to declassified: This transition can only be applied after approval from an authority competent to certify that the file does not have information that needs to be classified. Such authorities are called OCAs (see Section 4).

- Working paper to classified: This transition must also be approved by an OCA. In addition, a number of credentials must be issued by different officials to approve this authorization, and to decide the classified file's secrecy level, compartments, etc. These credentials are described in Section 4.4. The status of the file must be set to classified T T', where T is the time of the transition, and T' is determined by the OCA approving the transition.

- Classified to declassified: There is no need to explicitly apply this transition when the classification on a file expires, since the access policies (Section 3.3) automatically allow everyone read access after that time. The transition is needed only to prematurely declassify a file. In that case, approval from an OCA is needed.

3.3 Rules for Access to Files

Next we formalize in BL the highest level policy rules for file access. Access to a file depends on its state, and follows the informal guidelines described at the beginning of Section 3. We group our rules by the state to which they apply.

Default. In default state, a file may be read, written, and deleted only by its owner. This is captured by the following rules. The first rule states that it is the admin's policy that if file F is in default state (condition **has_xattr** F status default) and F is owned by K (condition **owner** K F), then K may read F. The second and third rules similarly allow K to write and delete F respectively. The term identity used in the third rule is the PCFS permission needed to delete a file (and also to rename it).[4]

```
admin says ((may K F read) :-
                has_xattr F status default,
                owner F K).
admin says ((may K F write) :-
                has_xattr F status default,
                owner F K).
```

[4]In contrast, in a POSIX compliant system, write permission on a file's parent directory suffices to delete or rename the file. The separate identity permission in PCFS allows access policies that are more fine grained, including the ones in this paper.

```
admin says  ((may K F identity) :-
                    has_xattr F status default,
                    owner F K).
```

Working Paper. If a file F is marked as a working paper at time T, then for 90 days after T, F may be read, written, or deleted at the discretion of the owner of file (which, as described in Section 3.2, may be an agency or group). This is enforced by the following rules. $90d$ denotes 90 days, and `is` X E is a special predicate that evaluates the expression E and unifies the result with X. The conditions K' says (`may` K F read), K' says (`may` K F write), and K' says (`may` K F identity) *delegate* authorization to K', the owner of file F.

```
admin says  (((may K F read) :-
                     has_xattr F status (working T),
                     owner F K',
                     K' says (may K F read),
                     is T' (T + 90d)) @ [T, T']).
admin says  (((may K F write) :-
                     has_xattr F status (working T),
                     owner F K',
                     K' says (may K F write),
                     is T' (T + 90d)) @ [T, T']).
admin says  (((may K F identity) :-
                     has_xattr F status (working T),
                     owner F K',
                     K' says (may K F identity),
                     is T' (T + 90d)) @ [T, T']).
```

It is instructive to observe the role of the @ connective in enforcing the 90 day restriction. For example, according to BL's inference rules, the consequence of the first rule is that for any time point $u \in [T, T+90d]$ at which **has_xattr** F status (working T), **owner** F K', and K' says (`may` K F read) all hold, admin says (`may` K F read) also holds. This is not the case if $u \notin [T, T+90d]$. If 90 days elapse since a file is made a working paper, none of the above rules allow any access to it. In that case, only the principal sysadmin has govern permission to the file (Section 3.2), and this principal must be asked to adjust the status of the file.

Classified. Read access to a classified file is based on properties of its classification such as its secrecy level, compartments, etc., as well as corresponding credentials of the principal to whom access is given. We capture these with the predicate `indi/has-clearances/file` K F which means that principal K's credentials suffice to allow it access to F. A large part of the paper is devoted to describing how this critical predicate is established; it is defined formally in Section 6. In addition to these properties and credentials, read access to a classified file is contingent on authorization from the file's owner. The following rule specifies this formally.

```
admin says (((may $K$ $F$ read) :-
        **has_xattr** $F$ status (classified $T$ $T'$),
        indi/has-clearances/file $K$ $F$,
        **owner** $F$ $K'$,
        $K'$ says (may $K$ $F$ read)) @ $[T, T']$).
```

The annotation @ $[T, T']$ restricts the scope of this rule to the duration for which the file is classified. After T', the file is readable by everyone (described next).

Declassified. A file is considered declassified if either its status is marked as such, or if the file is marked classified, but the classification has expired. In both cases, anyone may read the file. This is captured by the following rules.

```
admin says ((may $K$ $F$ read) :-
        **has_xattr** $F$ status declassified).

admin says (((may $K$ $F$ read) :-
        **has_xattr** $F$ status (classified $T$ $T'$)) @ $[T', \infty)$).
```

A consequence of the second rule is that if **has_xattr** F status (classified T T'), then for every time point $u \geq T'$, (admin says (may K F read)) @ $[u, u]$. This does not hold for $u < T'$.

3.3.1 Provisions for Counterintelligence Personnel

In addition to the above rules, there are provisions to allow counterintelligence personnel to read all files that may contain incriminating evidence against an individual they are investigating. Presumably, these provisions apply to files in all states. It is unclear how counterintelligence personnel are assigned to investigate individuals, and how it may be decided whether a file has incriminating evidence against a suspect or not. In our formalization we assume that a special principal oracle can determine these facts accurately. Formally, let the predicate `indi/is-ci` K K' mean that principal K is a counterintelligence officer investigating principal K', and let `indi/is-associated` K' F mean that file F may have incriminating evidence against the suspect principal K'. The following rule states that if the principal oracle states both these predicates, then K may read file F.

```
admin says ((may $K$ $F$ read) :-
        oracle says (indi/is-ci $K$ $K'$),
        oracle says (indi/is-associated $K'$ $F$)).
```

The principal oracle appears at many places in the rest of this paper. In each such case, it is assumed to assert relevant facts whose source is either unclear or unspecific from official documents.

4 File Classification

In Section 3.2 we stated that when a file is classified, a number of credentials must be issued to determine properties of the file such as its secrecy level, associated compartments, etc. This section explains these credentials in detail, as well as BL rules which combine these credentials to establish properties of a classified file. We start with an intuitive explanation of these properties, and subsequently present BL rules to establish them.

Briefly, there are three relevant properties of a classified file, each of which must be established before the file can be accessed (more precisely, these properties must be known in order to establish the predicate `indi/has-clearances/file` K F from Section 3.3):[5]

- Secrecy level: A secrecy level is an indicator of the sensitivity of the contents of a file. It is one of confidential, secret, or topsecret, in increasing order of sensitivity.[6] Read access to a classified file is restricted to individuals who have a secrecy clearance at a level equal to or greater than the secrecy level of the file.

- Citizenship requirement: A set of countries is associated with every classified file. Access is restricted only to citizens of those countries, and to those of the U.S. A commonly used abbreviation is "NOFORN" (no access to foreigners), which corresponds to an empty list of countries.

- Associated compartments: A compartment is a description of the purpose of a file, e.g., a project name or a division within the intelligence community. Every classified file is associated with zero or more compartments. Read access to a classified file is restricted only to those individuals who are associated with at least all compartments that the file is associated with.

Policies for giving clearances to individuals are discussed in Section 5. In this section we discuss rules pertaining to compartment creation and establishment of the properties listed above.

4.1 Original Classification Authorities

The authority to decide which file needs to be classified, and what secrecy level, citizenship requirements, and associated compartments a classified file will have rests with very high ranking officers of the executive branch of the government and their representatives. These individuals are called Original Classification Authorities or OCAs. We do not model formally how OCAs are determined. Instead, we assume that the principal oracle (introduced in Section 3.3) names OCAs. Let the predicate `indi/is-oca` O mean that principal O is an

[5] It is possible that there are other relevant properties in practice, but these three properties appear to be sufficiently representative.

[6] There is another secrecy level called sbu (sensitive but unclassified), or "for official use only". Files at this level are *not classified* – sbu is merely a directive to officials to be more careful than usual when handling such files. Therefore, we do not consider sbu in our formalization.

OCA. Then the following rule delegates authority over this predicate from admin to oracle.

```
admin  says  ((indi/is-oca O)  :-
                              oracle says (indi/is-oca O)).
```

4.2 Compartments

As mentioned earlier, a compartment describes the purpose of information it labels. For example, a compartment may be the name of an intelligence project. Files that have at least one compartment associated with them are called *compartmentalized* files. The purpose of associating a file with compartments is to restrict access to only those individuals who are affiliated with each of those compartments. In addition to restricting access, compartments associated with a classified file play a vital role in determining its secrecy level and citizenship requirements, as we discuss later in this section.

Compartment creation. A compartment is created by an OCA. The OCA also fixes several parameters that determine when an individual may be cleared into the compartment. Of these parameters, we model three prominent ones in this paper: (1) The minimum secrecy level at the which the individual must be cleared, (2) The minimum level of background check the individual must pass, and (3) Whether or not the individual has to pass a polygraph test. Formally we define the predicate compartment/is C L L' B to mean that C is a valid compartment (in practice, C is a unique string naming the compartment), clearance into which requires:

- A secrecy clearance at level L or higher. Secrecy clearances are described in Section 5.2.
- A background check equivalent to that needed for secrecy clearance at level L' or higher. Background checks are described in Section 5.1.
- A polygraph test if the boolean B is yes. Alternatively, if B is no, then a polygraph test is not necessary to be cleared into C. Polygraph tests are described in Section 5.1.

The following rule delegates the authority to create compartments from admin to every OCA O.

```
admin says  ((compartment/is C L L' B)  :-
                                  indi/is-oca O,
                                  O says (compartment/is C L L' B)).
```

SSO and SCG. When a compartment is created, an OCA appoints a special security officer (SSO) to manage the compartment. Afterwards, a set of guidelines for managing all information associated with the compartment is prepared. This set of guidelines is called the compartment's security classification guide (SCG); it must be approved by both an OCA and the SSO of the compartment to which the SCG pertains. Among, other things, the

SCG lays down procedures for deciding the secrecy level and citizenship requirements of any file associated with the compartment. As a result, when a file is classified, its associated compartments must be decided first, and subsequently its secrecy level and citizenship requirements must be determined using the SCGs of all the associated compartments.

In our formal model we abstract away the details of an SCG, and treat it only as a symbolic constant. Let the predicate `compartment/has-sso` C S mean that principal S is compartment C's special security officer, and let `compartment/has-scg` C SCG mean that SCG is the security classification guide of compartment C. Then, the first rule below allows an OCA O to assign an SSO S to a compartment C, while the second rule states that both an OCA and the SSO of compartment C must approve C's SCG.

```
admin says ((compartment/has-sso C S) :-
                                      indi/is-oca O,
                                      O says (compartment/has-sso C S)).
admin says ((compartment/has-scg C SCG) :-
                                      indi/is-oca O,
                                      O says (compartment/has-scg C SCG),
                                      compartment/has-sso C S,
                                      S says (compartment/has-scg C SCG)).
```

4.3 Establishing File Properties

Next, we discuss and formalize rules for determining a file's secrecy level, its citizenship requirements, and its associated compartments. As mentioned in Section 4.2, the compartments associated with a file must be decided first since they are necessary to authorize the file's secrecy level and citizenship requirements.

Determining a file's associated compartments. Let the predicate `file/has-compartments` F CL mean that file F is associated with exactly the compartments in the list CL. As per official guidelines, establishing this predicate requires two kinds of approvals: (a) an approval from an OCA stating that this should be the case, and (b) approvals from the SSOs of all compartments in the list CL stating that the file may be associated with all the compartments in CL. Modeling the second requirement in BL is slightly tricky; we use a recursively defined helper predicate `file/has-compartments/h` F CL CL' which means that the SSOs of all compartments in CL' agree that F should be associated with all compartments in CL. The following rule uses this predicate with $CL' = CL$ to allow a file to be associated with a list of compartments CL.

```
admin says ((file/has-compartments F CL) :-
                                        indi/is-oca O,
                                        O says (file/has-compartments F CL),
                                        file/has-compartments/h F CL CL).
```

The following two rules define the helper predicate `file/has-compartments/h` F CL CL' by induction on CL'. The symbol nil denotes the empty list and | is an infix cons constructor.

admin says (`file/has-compartments/h` F CL nil).

admin says ((`file/has-compartments/h` F CL $(C' \mid CL')$) :-
`compartment/has-sso` C' S,
S says (`file/has-compartments` F CL),
`file/has-compartments/h` F CL CL').

The second rule above means that admin will believe that the SSOs of all compartments in $C' \mid CL'$ agree that F should be associated with the compartments in CL if (a) The SSO S of compartment C' agrees to this fact (first two conditions of the rule) and (b) Recursively, the SSOs of all compartments in CL' agree to this fact (third condition).

Determining a file's secrecy level. As per official guidelines, a file's secrecy level may be set to L if: (a) an OCA says that this should be case, and (b) the SSOs of all compartments associated with the file agree that the SCGs of their respective compartments allow the file to be given secrecy level L. Formally, let the predicate `file/has-level` F L mean that file F has secrecy level L, and `file/has-level/h` F L CL mean that the SSOs of all compartments in CL agree that F may be given secrecy level L in accordance with their respective SCGs. Then the following rule formally captures the above conditions for giving the secrecy level L to file F.

admin says ((`file/has-level` F L) :- `indi/is-oca` O,
O says (`file/has-level` F L),
`file/has-compartments` F CL,
`file/has-level/h` F L CL).

The following two rules define the predicate `file/has-level/h` F L CL defined by induction on CL. The predicate `file/has-level/scg` F L SCG is intended to mean that the security classification guide SCG mandates that file F be given secrecy level L.

admin says (`file/has-level/h` F L nil).

admin says ((`file/has-level/h` F L $(C' \mid CL')$) :-
`compartment/has-sso` C' S,
`compartment/has-scg` C' SCG,
S says (`file/has-level/scg` F L SCG),
`file/has-level/h` F L CL').

According to the second rule above, admin believes that the SSOs of all compartments in $C' \mid CL'$ agree that F should have secrecy level L if (a) the SSO S of C' states that this assignment of level would be in accordance with the SCG of C' (third condition of the rule), and (b) Recursively, the SSOs of all compartments in CL' agree with this assignment (fourth condition).

It follows from these rules that if there are no compartments associated with a file F, i.e., if admin says (`file/has-compartments` F nil), then an OCA O's statement O says (`file/has-level` F L) suffices to give a security level L to a file.

Determining a file's citizenship requirements. Determining the citizenship requirements for reading a file is similar to determining the file's secrecy level – an OCA must approve the list of countries to whose citizens access should be restricted, and the SSOs of all compartments associated with the file must certify that this list would be allowed by their respective SCGs. Formally, let the predicate `file/has-citizenship` F UL mean that reading file F requires a citizenship of one of the countries in UL (or of the U.S.), `file/has-citizenship/h` F UL CL mean that the SSOs of all compartments in CL agree with this requirement, and `file/has-citizenship/scg` F UL SCG mean that SCG approves this requirement. Then the following three rules may used to determine a file's citizenship requirements.

```
admin says ((file/has-citizenship F UL) :-
                                    indi/is-oca O,
                                    O says (file/has-citizenship F UL),
                                    file/has-compartments F CL,
                                    file/has-citizenship/h F UL CL).

admin says (file/has-citizenship/h F UL nil).

admin says ((file/has-citizenship/h F UL (C′ | CL′)) :-
                                    compartment/has-sso C′ S,
                                    compartment/has-scg C′ SCG,
                                    S says (file/has-citizenship/scg F UL SCG),
                                    file/has-citizenship/h F UL CL′).
```

As in the case of rules for determining secrecy levels, if there are no compartments associated with a file F, i.e., if admin says (`file/has-compartments` F nil), then an OCA O's statement O says (`file/has-citizenship` F UL) suffices to give a citizenship requirement UL to a file.

4.4 Summary of File Classification

As mentioned in Section 3.2, before setting a file F's status attribute to classified T T', the principal sysadmin must ensure that enough credentials are in place to determine the file's secrecy level, citizenship requirements, and associated compartments. The credentials required follow from the rules discussed in Section 4.3, and are summarized below. Although not formalized here, T and T' must also be obtained from an OCA.

- Credentials to determine associated compartments CL
  - An OCA O must issue the credential O says (`file/has-compartments` F CL).
  - For every compartment $C \in CL$, the SSO S of C must issue the credential S says (`file/has-compartments` F CL).

- Credentials to determine secrecy level L
  - An OCA O must issue the credential O says (`file/has-level` F L).
  - For every compartment $C \in CL$, where CL is the list from the previous point, the SSO S of C must issue the credential S says (`file/has-level/scg` F L SCG), where SCG is the security classification guide of C.
- Credentials to determine citizenship requirements UL
  - An OCA O must issue the credential O says (`file/has-citizenship` F UL).
  - For every compartment $C \in CL$, where CL is the list from the first point, the SSO S of C must issue the credential S says (`file/has-citizenship/scg` F UL SCG), where SCG is the security classification guide of C.

In practice, any issued credential will be valid for only a stipulated duration of time. This gets represented in BL through the @ connective. For example, if an OCA O says that file F should have secrecy level L from 2009 to 2011, this would be represented in BL as (O says (`file/has-level` F L)) @ $[2009, 2011]$. In general, all credentials in the list above may be time-restricted using @ connectives. BL's inference rules propagate these time restrictions to other facts derived from the credentials and policy rules.

5 Individual Clearances

Individuals require clearance both at secrecy levels and into compartments, as well as citizenship of specific countries to read classified files. We call these three primary clearances of individuals. In order to obtain primary clearances, other auxiliary clearances are needed. These include polygraph tests and background checks. In this section we formalize the methods for obtaining auxiliary clearances, as well as rules for combining them to determine clearance at secrecy levels and into compartments. We start with the auxiliary clearances.

5.1 Auxiliary Clearances

Polygraph clearance. Individuals may need to pass a polygraph test to get clearance into certain compartments (Sections 4.2 and 5.2). Polygraph tests can be administered and certified by trained individuals, whom we call polygraph administrators. The procedures for identifying polygraph administrators are beyond the scope of our formalization; we simply assume that oracle names polygraph administrators. Let `indi/is-polygraph-admin` PA mean that principal PA is a trusted polygraph administrator, and let `indi/has-polygraph` K mean that principal K has passed a polygraph test. The following rule states that if oracle says that PA is a polygraph administrator, and PA says that K has passed a polygraph test, then admin will believe the latter.

```
admin says ((indi/has-polygraph K) :-
                                              oracle says (indi/is-polygraph-admin PA),
                                              PA says (indi/has-polygraph K)).
```

Background checks. A background check certifies an individual's past. It is necessary to get clearance both at secrecy levels and into compartments. There are two commonly used background checks: (1) National Agency Check with Local Agency Check and Credit Check or NACLC, and (2) Single Scope Background Investigation or SSBI. NACLC is an investigation of an individual's criminal records and credit history. SSBI includes the NACLC and in addition requires interviews of colleagues and investigation of family history. We assume that certain principals called *background administrators* are certified to check others' backgrounds. Background administrators are assumed to be determined by the principal oracle.

From the perspective of formalization, it is very convenient to abstract background checks by the secrecy level for which they are mandatory. Informally speaking, for example, a *background check at level* confidential would correspond to a background check that is needed to get clearance at secrecy level confidential. This kind of an abstraction is useful because, as per official guidelines, background checks conducted for clearance at secrecy levels expire at fixed intervals of time, and a similar expiration applies to other applications of background checks (e.g., for clearance into compartments). The actual check corresponding to each secrecy level and its expiration time is shown in the table below.

Abstract level of background check	Actual background check needed and expiration
confidential	NACLC, expires in 15 years
secret	NACLC, expires in 10 years
topsecret	SSBI, expires in 5 years

Let `indi/is-background-admin` BA mean that principal BA is a background administrator. Further let `indi/has-naclc` K T mean that principal K passed an NACLC at time T,[7] and `indi/has-ssbi` K T mean that principal K passed an SSBI at time T. The following rules define the predicate `indi/has-background` K L, which means that principal K has a background check that is needed for clearance at secrecy level L. There are three rules, one for each possible value of L. A salient point to observe is the use of the @ connective for automatically expiring background checks in accordance with the table above. The symbol y following a number means "years". Hence $15y$ means 15 years. As an example, the first rule below means that if oracle states that BA is a background administrator and BA states that K passed an NACLC at time T, then admin believes that K has a background check at level confidential in the interval $[T, T + 15y]$.

```
admin says (((indi/has-background K confidential) :-
                              oracle says (indi/is-background-admin BA),
                              BA says (indi/has-naclc K T),
                              is T' (T + 15y)) @ [T, T']).
```

[7] In practice, the NACLC for secret clearance may be more extensive than the NACLC for confidential clearance. Even if there is such a distinction, we blur it in our formalization.

admin says $(((\texttt{indi/has-background}\ K\ \mathsf{secret})$:-
oracle says $(\texttt{indi/is-background-admin}\ BA)$,
BA says $(\texttt{indi/has-naclc}\ K\ T)$,
$\texttt{is}\ T'\ (T + 10y))$ @ $[T, T'])$.

admin says $(((\texttt{indi/has-background}\ K\ \mathsf{topsecret})$:-
oracle says $(\texttt{indi/is-background-admin}\ BA)$,
BA says $(\texttt{indi/has-ssbi}\ K\ T)$,
$\texttt{is}\ T'\ (T + 5y))$ @ $[T, T'])$.

The remaining policy rules refer only to the predicate `indi/has-background` $K\ L$, not to the predicates `indi/has-naclc` $K\ T$ and `indi/has-ssbi` $K\ T$.

5.2 Primary Clearances

An individual's clearance at a secrecy level, clearance into compartments, as well as citizenship directly determine what classified files she has access to. We now describe rules that define how these are determined.

Citizenship. We assume that oracle decides the citizenship of each individual. Let `indi/has-citizenship` $K\ U$ mean that principal K is a citizen of country U. The following rule delegates authority over this predicate from admin to oracle.

admin says $((\texttt{indi/has-citizenship}\ K\ U)$:-
oracle says $(\texttt{indi/has-citizenship}\ K\ U))$.

A useful, related predicate is `indi/has-citizenship/list` $K\ UL$, which means that K is a citizen of at least *one of the countries* in the list UL. The following two rules define this predicate by induction on the list UL.

admin says $((\texttt{indi/has-citizenship/list}\ K\ (U \mid UL))$:-
$\texttt{indi/has-citizenship}\ K\ U)$.

admin says
$((\texttt{indi/has-citizenship/list}\ K\ (U \mid UL))$:-
oracle says $(\texttt{indi/has-citizenship/list}\ K\ UL))$.

Clearance at secrecy levels. As mentioned in Section 5.1, an individual must pass a background check at level L in order to get clearance at secrecy level L. In addition, the individual must have a *need to get the clearance*. Since the factors determining this need are varied and are not completely specified, we simply assume here that oracle may assert this need. Let `indi/has-level` $K\ L$ mean that individual K has clearance at secrecy level L, and `indi/needs-level` $K\ L$ mean that principal K has a need to get clearance at secrecy level L. `level/below` $L\ L'$ means that level L is below the level L' in the order confidential < secret < topsecret. It is defined later. The following rule states that admin will believe that K has clearance at secrecy level L if oracle says that K needs this clearance, and K has passed a background check at some level L' which is higher than L.

```
admin says ((indi/has-level K L) :-
                         oracle says (indi/needs-level K L),
                         indi/has-background K L',
                         level/below L L').
```

As formalized in Section 5.1, the validity of `indi/has-background` K L' is limited to 15, 10, or 5 years depending on L'. The above rule and the inference rules of BL transfer the same restrictions to `indi/has-level` K L. The predicate `level/below` is defined by the rules below. Since it is reasonable to assume that the order it defines is believed by all principals, these rules are stated by the strongest principal ℓ (According to BL's inference rules, (ℓ says s) $\supset$ (k says s) for every k and s).

```
ℓ says (level/below L L).

ℓ says (level/below confidential secret).

ℓ says (level/below secret topsecret).

ℓ says (level/below confidential topsecret).
```

Clearance into compartments. As mentioned in Section 4.2, to be cleared into a compartment, an individual must satisfy all its requirements – secrecy level, background check, and a polygraph test if needed. These requirements are uniquely determined from the predicate `compartment/is` C L L' B, which is established when the compartment C is created. Let the predicates `indi/has-comp-level` K C, `indi/has-comp-background` K C, and `indi/has-comp-polygraph` K C mean that an individual has clearance at an appropriate secrecy level, background check, and polygraph check (if needed) for being cleared into compartment C. The following rules define these predicates by considering respectively the 2nd, 3rd, and 4th arguments of the predicate `compartment/is` C L L' B. An underscore _ represents an implicitly named variable, whose instantiated value is irrelevant to the rule.

```
admin says ((indi/has-comp-level K C) :-
                              compartment/is C L _ _,
                              indi/has-level K L'',
                              level/below L L'').
admin says ((indi/has-comp-background K C) :-
                                   compartment/is C _ L' _,
                                   indi/has-background K L'',
                                   level/below L' L'').
admin says ((indi/has-comp-polygraph K C) :-
                                  compartment/is C _ _ yes,
                                  indi/has-polygraph K).
admin says ((indi/has-comp-polygraph K C) :-
                                  compartment/is C _ _ no).
```

Using the above predicates, we define the predicate `indi/has-compartment` K C which means that an individual K is cleared into the compartment C. An important fact to observe here is that in addition to satisfying the three requirements of the compartment, the SSO S of the compartment must certify the clearance, and, as in the case of clearance at secrecy levels, the principal oracle must certify that the principal actually needs the clearance (predicate `indi/needs-compartment` K C).

```
admin says ((indi/has-compartment K C) :-
                                oracle says (indi/needs-compartment K C)
                                compartment/has-sso C S,
                                S says (indi/has-compartment K C),
                                indi/has-comp-level K C,
                                indi/has-comp-background K C,
                                indi/has-comp-polygraph K C).
```

Finally, the following two rules define a related, useful predicate `indi/has-compartment/list` K CL which means that K is cleared into *all compartments* in the list CL.

```
admin says (indi/has-compartment/list K nil).

admin says ((indi/has-compartment/list K (C | CL)) :-
                                    indi/has-compartment K C,
                                    indi/has-compartment/list K CL).
```

5.3 Summary of Individual Clearances

We close this section with a summary of credentials needed to give various clearances to an individual K.

- Credentials to establish polygraph clearance
  - A polygraph administrator PA must issue the credential PA says (`indi/has-polygraph` K)
- Credentials to certify background check at level L
  - If L is confidential or secret, then a background administrator BA must issue the credential BA says (`indi/has-naclc` K T). The check is valid for 15 years after T if $L =$ confidential and for 10 years after T if $L =$ secret.
  - If L is topsecret, then a background administrator BA must issue the credential BA says (`indi/has-ssbi` K T). The check is valid for 5 years after T.
- Credentials to determine citizenship of country U
  - oracle must issue the credential oracle says (`indi/has-citizenship` K U).

- Credentials for secrecy clearance at level L
  - oracle must issue the credential oracle says (`indi/needs-level` K L)
  - Credentials to certify background check at level L or higher as determined by the second point above.
- Credentials for clearance into compartment C established with the predicate `compartment/is` C L L' B
  - oracle must issue the credential oracle says (`indi/needs-compartment` K C).
  - The SSO S of C must issue the credential S says (`indi/has-compartment` K C).
  - Credentials for secrecy clearance at level L or higher as determined by the fourth point above.
  - Credentials to certify background check at level L' or higher as determined by the second point above.
  - Credentials to establish polygraph clearance if $B =$ yes as determined by the first point above.

6 Clearances to Classified Files

In Section 3.3 we introduced the predicate `indi/has-clearances/file` K F, which means that principal K has enough clearance to read classified file F. Building on other predicates defined in Sections 4 and 5, we now provide rules that define this critical predicate.

First, we define three auxiliary predicates using the fairly straightforward rules below: (a) `indi/has-level/file` K F, which means that principal K has clearance at a secrecy level higher than that of file F, (b) `indi/has-comps/file` K F, which means that principal K is cleared into all compartments that F is associated with, and (c) `indi/has-cit/file` K F, which means that principal K is a citizen of at least one country in the citizenship requirements of F.

```
admin says ((indi/has-level/file K F) :-
                                    file/has-level F L,
                                    indi/has-level K L',
                                    level/below L L').
admin says ((indi/has-comps/file K F) :-
                                    file/has-compartments F CL,
                                    indi/has-compartment/list K CL).
admin says ((indi/has-cit/file K F) :-
                                  file/has-citizenship F UL,
                                  indi/has-citizenship/list K UL).
admin says ((indi/has-cit/file K F) :-
                                  indi/has-citizenship K usa).
```

The last rule means that any U.S. citizen satisfies the citizenship requirement for reading a file, irrespective of the latter's actual citizenship requirements. The following rule defines the predicate `indi/has-clearances/file` K F using these three predicates.

```
admin says ((indi/has-clearances/file K F) :-
                                   indi/has-level/file K F,
                                   indi/has-comps/file K F,
                                   indi/has-cit/file K F).
```

7 Conclusion

We have formalized in the logic BL common rules for access to classified information, and discussed how they may be enforced in PCFS. The formalization relies on several features of BL including its support for explicit time and system state. While this demonstrates the expressiveness of BL and PCFS as a framework for enforcement of complex access policies, a significant question that remains open is enforcement of rules for changing system state, such as those discussed in Section 3.2. We believe that both programs and workflows that perform such transitions may be subject to rigorous analysis through extensions of authorization logics, such as those in recent work [5].

References

[1] Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In G. Tsudik, editor, *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 52–62, Singapore, November 1999. ACM Press.

[2] Lujo Bauer. *Access Control for the Web via Proof-Carrying Authorization.* PhD thesis, Princeton University, November 2003.

[3] Lujo Bauer, Scott Garriss, Jonathan M. McCune, Michael K. Reiter, Jason Rouse, and Peter Rutenbar. Device-enabled authorization in the Grey system. In *Information Security: 8th International Conference (ISC '05)*, Lecture Notes in Computer Science, pages 431–445, September 2005.

[4] Henry DeYoung, Deepak Garg, and Frank Pfenning. An authorization logic with explicit time. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF-21)*, Pittsburgh, Pennsylvania, June 2008. IEEE Computer Society Press. Extended version available as Technical Report CMU-CS-07-166.

[5] Henry DeYoung and Frank Pfenning. Reasoning about the consequences of authorization policies in a linear epistemic logic. In *Workshop on Foundations of Computer Security (FCS)*, 2009. To appear.

[6] Deepak Garg. Proof search in an authorization logic. Technical Report CMU-CS-09-121, Carnegie Mellon University, 2009.

[7] Deepak Garg and Frank Pfenning. Non-interference in constructive authorization logic. In J. Guttman, editor, *Proceedings of the 19th Computer Security Foundations Workshop (CSFW '06)*, pages 283–293, Venice, Italy, July 2006. IEEE Computer Society Press.

[8] Deepak Garg and Frank Pfenning. A proof-carrying file system. Technical Report CMU-CS-09-123, Carnegie Mellon University, 2009.

[9] Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, November 1992.

[10] Office of the Director of Central Intelligence. DCID 1/19: Security policy for sensitive compartmented information and security policy manual, 1995. Online at `http://www.fas.org/irp/offdocs/dcid1-7.html`.

[11] Office of the Director of Central Intelligence. DCID 1/7: Security controls on the dissemination of intelligence information, 1998. Online at `http://www.fas.org/irp/offdocs/dcid1-19.html`.

[12] Office of the Press Secretary of the White House. Executive order 12958: Classified national security information, 1995. Online at `http://nsi.org/Library/Govt/ExecOrder12958.html`.

[13] Office of the Press Secretary of the White House. Executive order 13292: Further amendment to executive order 12958, as amended, classified national security information, 2003. Online at `http://nodis3.gsfc.nasa.gov/displayEO.cfm?id=EO_13292_`.

A Summary of Predicates Used in the Formalization

The following table lists all predicates used in this paper, the sections of the paper in which they are described, and their intuitive meanings.

Predicate	Section	Meaning
`compartment/has-scg` C SCG	4.2	SCG is compartment C's security classification guide
`compartment/has-sso` C S	4.2	S is compartment C's special security officer (SSO)
`compartment/is` C L L' B	4.2	C is a compartment, clearance into which requires secrecy clearance at level L, background check at level L', and a polygraph test if $B = \mathsf{yes}$.
`file/has-citizenship` F UL	4.3	Read access to file F is restricted to citizens of countries in the list UL (and of the U.S.)
`file/has-citizenship/h` F UL CL	4.3	The SSOs of all compartments in the list CL certify that read access to F should be restricted to citizens of countries in the list UL (and of the U.S.)

`file/has-citizenship/scg` F UL SCG	4.3	It is conformant with SCG that read access to file F be restricted to citizens of countries in the list UL (and of the U.S.)
`file/has-compartments` F CL	4.3	File F is associated with all compartments in the list CL
`file/has-compartments/h` F CL CL'	4.3	The SSOs of all compartments in the list CL' certify that is okay to associate file F with all compartments in the list CL
`file/has-level` F L	4.3	File F has secrecy level L
`file/has-level/h` F L CL	4.3	The SSOs of all compartments in the list CL certify that is okay to give file F secrecy level L
`file/has-level/scg` F L SCG	4.3	It is conformant with SCG that file F have secrecy level L
has_xattr F A V	3.1	The extended attribute named A on file F is set to value V
`indi/has-background` K L	5.1	Principal K has a background check which is mandatory for clearance at secrecy level L
`indi/has-citizenship` K U	5.2	Principal K is a citizen of country U
`indi/has-citizenship/list` K UL	5.2	Principal K is a citizen of at least one of the countries in the list UL
`indi/has-cit/file` K F	6	Principal K has the citizenship of one of the countries associated with file F (or of the U.S.)
`indi/has-clearances/file` K F	6	Principal K has enough security clearances to read classified file F
`indi/has-compartment` K C	5.2	Principal K is cleared into compartment C
`indi/has-compartment/list` K CL	5.2	Principal K is cleared into all compartments in the list CL
`indi/has-comp-background` K C	5.2	Principal K has passed a background check sufficient for clearance into compartment C
`indi/has-comp-level` K C	5.2	Principal K has secrecy clearance at a level higher than that needed for clearance into compartment C
`indi/has-comp-polygraph` K C	5.2	If clearance into compartment C requires a polygraph test, then principal K has passed one
`indi/has-comps/file` K F	6	Principal K is cleared into all compartments associated with file F
`indi/has-level` K L	5.2	Principal K is cleared at secrecy level L
`indi/has-level/file` K F	6	Principal K has secrecy clearance at a level equal to or above that of file F
`indi/has-naclc` K T	5.1	Principal K passed an NACLC at time T
`indi/has-polygraph` K	5.1	Principal K passed a polygraph test
`indi/has-ssbi` K T	5.1	Principal K passed an SSBI at time T

`indi/is-associated` K F	3.3.1	File F may potentially have incriminating evidence against principal K
`indi/is-background-admin` BA	5.1	Principal BA is certified to check others' backgrounds
`indi/is-ci` K K'	3.3.1	Principal K is a counterintelligence officer who is investigating principal K'
`indi/is-oca` O	4.1	Principal O is an Original Classification Authority (OCA)
`indi/is-polygraph-admin` PA	5.1	Principal PA is certified to administer polygraph tests on others
`indi/needs-compartment` K C	5.2	Principal K needs clearance into compartment C
`indi/needs-level` K L	5.2	Principal K needs clearance at secrecy level L
`level/below` L L'	5.2	Secrecy level L is below L' ($\mathsf{confidential} < \mathsf{secret} < \mathsf{topsecret}$)
`may` K F P	3.3	Principal K has permission P on file F
owner F K	3.3	File F is owned by principal K