IS 2620

Formal Verification/Methods
Common Criteria

Lecture 11

Oct 25, 2018

Formal Verification

- Formal verification relies on
  - Descriptions of the properties or requirements
  - Descriptions of systems to be analyzed, and
  - Verification techniques showing requirements are met by system description
    - Rely on underlying mathematical logic system and the proof theory of that system

Formal Approach

- Formal Models use language of mathematics
  - Specification languages
    - For policies, models and system descriptions
    - Well-defined syntax and semantics – based on maths
- Current trends - two general categories
  - Inductive verification techniques
  - Model checking techniques
    - Differences based on
      - Intended use, degree of automation, underlying logic systems, etc.

Verification techniques – Criteria for classifying verification technologies

- Proof-based vs model-based
  - Proof-based
    - Formula define premises : embody the system description
    - Conclusions: what needs to be proved
  - Proof shows how to reach conclusions from premises
    - Intermediate formulas need to be found to reach conclusions
  - Model-based:
  - Premises and conclusions have/exhibit the same truth table values
- Degree of automation
  - manual or automated (degree) & in between

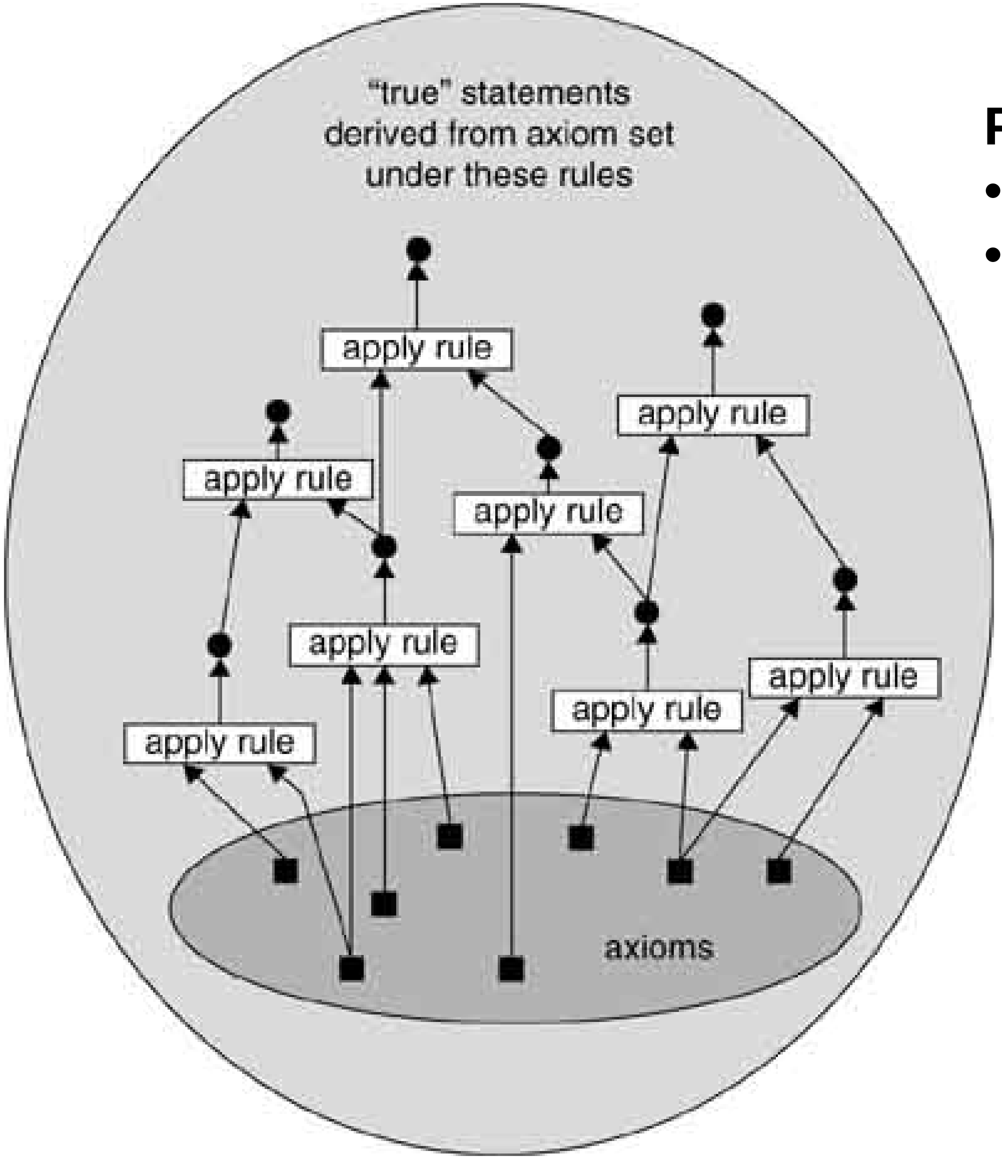
Verification techniques – Criteria for classifying verification technologies

- Full verification vs property verification
  - Does methodology model full system?
  - Or just prove certain key properties?
    - Examples?
- Intended domain of application
  - HW/SW, sequential or concurrent, reactive or terminating, ..
- Predevelopment vs post development
  - As design aid or after design has been completed

Inductive verification

- Typically more general
- May be used
  - To find flaws in design
  - To verify the properties of computer programs
- Uses theorem provers
  - E.g., uses predicate/propositional calculus
  - A sequence of proof steps starting with premises of the formula and eventually reaching a conclusion

Propositional logic



Propositional

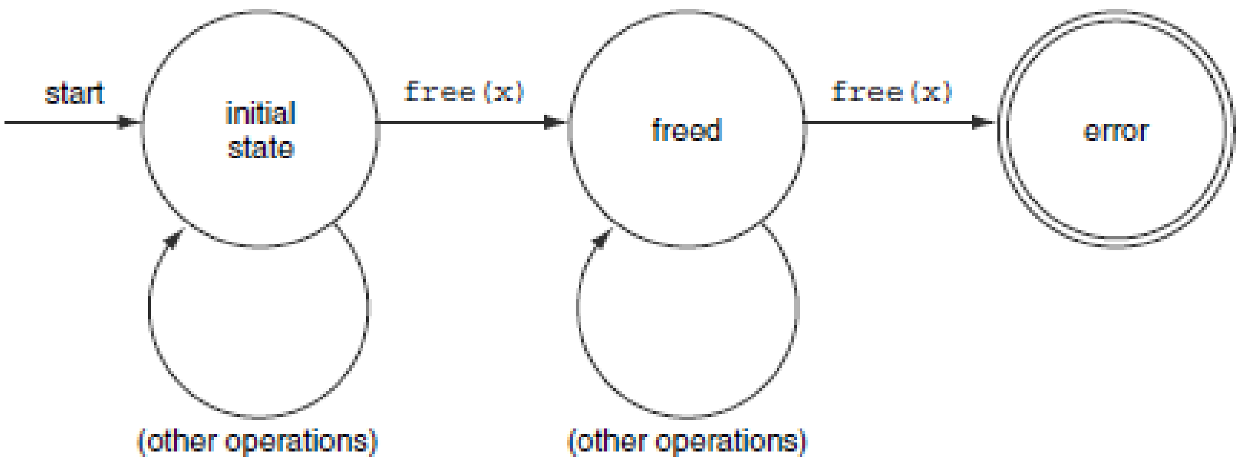
- Axioms
- Inference rules

Boolean

- And
- Or
- Not
- Implies

Model-checking

- Systems modeled as state transition systems
  - Formula may be true in some states and false in others
  - Formulas may change values as systems evolve
- Properties are formulas in logic
  - Truth values are dynamic (Temporal logic)
- Show: Model and the desired properties are semantically equivalent
  - Model and properties express the same truth table
- Often used after development is complete but before a product is released to the general market
  - Primarily for reactive, concurrent systems



Developed primarily for concurrent/reactive systems that react to environment

Formal Verification: Components

- Formal Specification
  - Defined in unambiguous (mathematical) language – precise semantics!
  - Restricted syntax, and well-defined semantics based on established mathematical concepts
    - Example: BLP Model
- Implementation Language
  - Generally somewhat constrained
- Formal Semantics relating the two
- Methodology to ensure implementation ensures specifications met

A *formal specification* is a specification written in a formal language with a restricted syntax and well-defined semantics based on well-established mathematical concepts.

Specification Languages

- Specify WHAT, not HOW
  - Valid states of system
  - Pre/Post-conditions of operations
- Non-Procedural
- Typical Examples:
  - Propositional / Predicate Logic
  - Temporal Logic (supports before/after conditions)
  - Set-based models
    - E.g., RBAC, formal Bell-LaPadula

Example: Primitive commands (HRU)

Create subject s	Creates new row, column in ACM s does not exist prior to this
Create object o	Creates new column in ACM o does not exist prior to this
Enter r into $a[s, o]$	Adds r right for subject s over object o Ineffective if r is already there
Delete r from $a[s, o]$	Removes r right from subject s over object o
Destroy subject s	Deletes row, column from ACM;
Destroy object o	Deletes column from ACM

Example: Primitive commands (HRU)

Create subject s

Creates new row, column in ACM; s does not exist prior to this

Precondition: $s \notin S$

Postconditions:

$S' = S \cup \{ s \}, O' = O \cup \{ s \}$

$(\forall y \in O')[a'[s, y] = \varnothing]$ **(row entries for s)**

$(\forall x \in S')[a'[x, s] = \varnothing]$ **(column entries for s)**

$(\forall x \in S)(\forall y \in O)[a'[x, y] = a[x, y]]$

Safety Theorems

Specification Languages

- Must support machine processing
  - Strong typing
  - Model input/output/errors
- Example: SPECIAL (from SRI)
  - First order logic based; Non procedural
  - Strongly typed
  - Expressive; has capability to describe
    - Inputs, constraints, errors, outputs
    - A rich set of built-in operators

SPECIAL has a rich set of built-in operators, including set operations such as UNION and DIFF; logical operators such as AND, OR, and => (implies); universal and existential quantifiers (FORALL, EXISTS); IF/THEN/ELSE constructs; arithmetic operators; and many others.

Well suited for functional specification

SPECIAL

- Specification modules for a system
  - Specifier defines the scope of the module
  - Provides convenience and ease of manipulation
- Sections for describing
  - Types,
    - E.g., DESIGNATOR type: Allows use of type whose specifics are to be defined at a lower level of abstraction
  - Parameters: Constants and entities
  - Assertions
    - About elements in the module
  - Functions – heart of SPECIAL
    - Statement variables and state transitions
    - Private or visible outside the module

VFUN: describes functions that return a value (state)
OFUN/OVFUN: describe state transitions

Example: SPECIAL

```
MODULE Bell_LaPadula_Model Give_access

TYPES

Subject_ID:  DESIGNATOR;
Object_ID:   DESIGNATOR;
Access_Mode: {OBSERVE_ONLY, ALTER_ONLY, OBSERVE_AND_ALTER};
Access:      STRUCT_OF( Subject_ID    subject;
                        Object_ID     object;
                        Access_Mode   mode);

FUNCTIONS

VFUN active (Object_ID object) -> BOOLEAN active:
HIDDEN;
INITIALLY
      TRUE;

VFUN access_matrix () -> Accesses accesses:
HIDDEN;
INITIALLY
      FORALL Access a: a INSET accesses => active (a.object);

OFUN give_access(Subject_ID giver; Access  access);
ASSERTIONS
      active(access.object) = TRUE;
EFFECTS
      'access_matrix() = access_matrix() UNION (access);

END_MODULE
```

Example: Enhanced Hierarchical Development Methodology

- Based on HDM
  - A general purpose design and implementation methodology
  - Goal was
    - To mechanize and formalize the entire development process
      - For reliable, verifiable and maintainable software
    - Design specification and verification + implementation specification and verification
      - Key idea; Successive refinement of specification
  - Design Specification:
    - hierarchy of abstract machines with increasing levels of details
- Proof-based method
  - Uses Boyer-Moore Theorem Prover

Levels of Abstraction

Requirements

The requirements are analyzed and accepted

Model

The model is proven to be internally consistent and is used as a basis for verification of the lower abstract machines

External interfaces AM1

The first abstract machine is generally the external interface specification, often called a Top Level Specification (TLS) or Formal TLS (FTLS)

External interfaces AM2

Each abstract machine is mapped to successively lower-level machines, which represent successively lower levels of specification of the system

External interfaces AM*n*

The lowest-level specification id the so-called primitive machine, which is some combination of hardware and software on which the verified system runs

Hierarchy Specification Language for *hierarchy speciation*

Abstract machines in SPECIAL

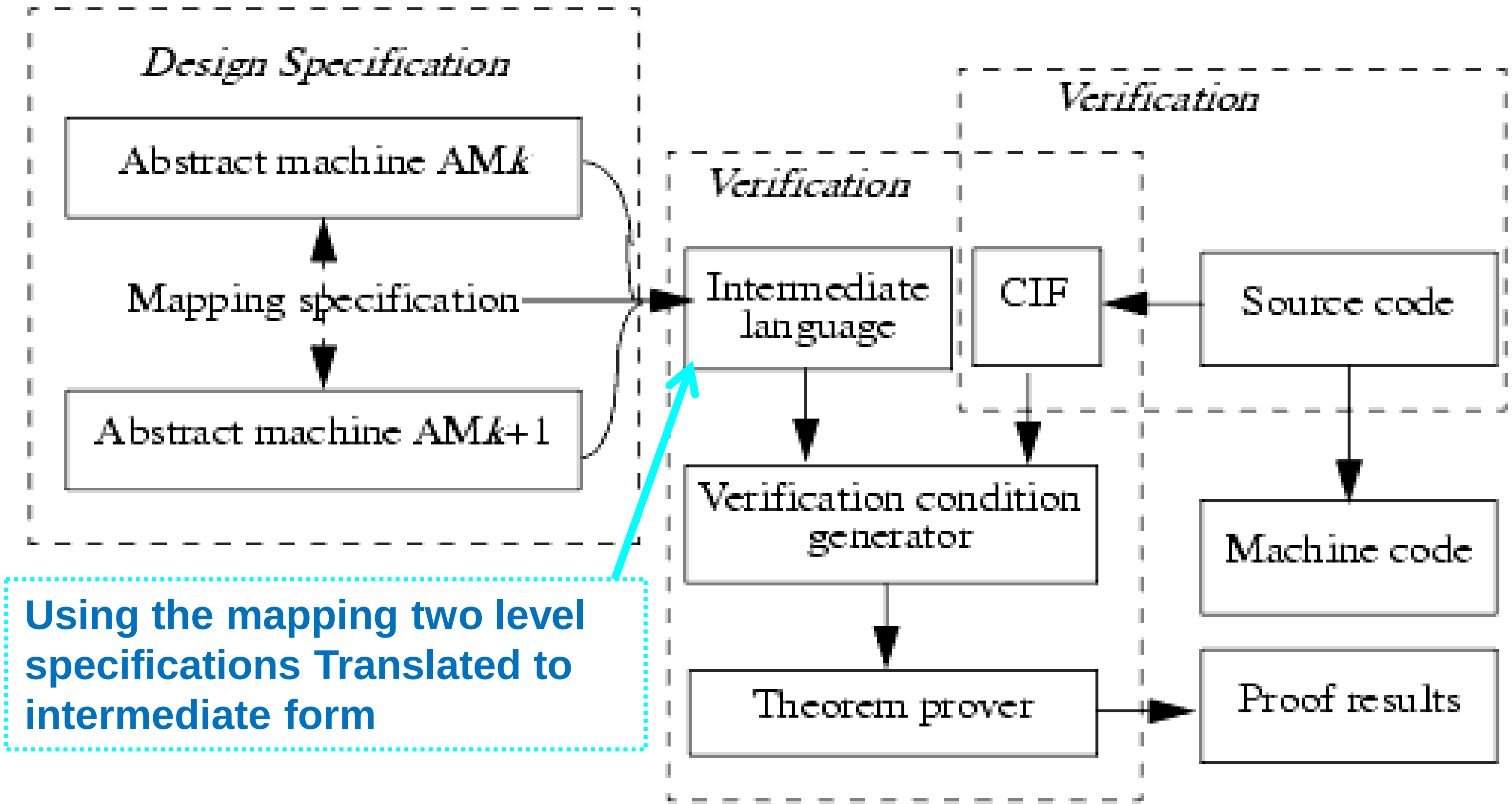
HDM Module and Mapping specification in SPECIAL

Example: Enhanced Hierarchical Development Methodology

- Hierarchical approach
  - *Abstract Machines* defined at each level
    - Hierarchy specification in in Hierarchy Specification Language (HSL)
    - AM specification written in SPECIAL
  - *Mapping Specifications* in SPECIAL
    - define functionality in terms of machines at next lower layer
  - *Hierarchy Consistency Checker*
    - validates consistency of HS, Module Spec and Mapping Spec
- Compiler: programs for each AM in terms of calls to lower level
  - that maps a program into a Common Internal Form (CIF) for HDM tools
  - Two levels of spec translated to CIF → correctness is verified (BMT prover)
- Successfully used on MLS systems
  - Few formal policy specifications outside MLS domain

HDM Verification

Used for MLS



Boyer-Moore Theorem Prover

- Fully automated
  - No interface for commands or directions
  - User provides all the theorems, axioms, lemmas, assertions
    - LISP like notation
  - Very difficult for proving complex theorems
- Key idea
  - Used extended propositional calculus
  - Efficiency – to find a proof.

Boyer-Moore Theorem Prover

- Steps:
  - *Simplify* the formula
    - Apply axioms, lemmata, theorems
  - *Reformulate* the formula with equivalent terms
    - E.g., replace x-1, x by y and y+1
  - *Substitute* equalities
  - *Generalize* the formula by introducing variables
  - *Eliminate* irrelevant terms
  - *Induct* to prove

Gypsy verification environment (GVE)

- Based on Pascal
  - Formal proof and runtime validation support
  - Focused on Implementation proofs rather than design proofs
    - verification of specification and its implementation
  - Also to support incremental development
- Specifications defined on procedures
  - Entry conditions, Exit conditions, Assertions
- Proof techniques ensure exit conditions / assertions met given entry conditions
  - Also run-time checking

Other Examples

- Prototype Verification System (PVS)
  - Based on EHDM
  - Interactive theorem-prover
- Symbolic Model Verifier
  - Temporal logic based / Control Tree Logic
  - Notion of “path” – program represented as tree
  - Statements that condition must hold at *a* future state, *all* future states, all states on one path, etc.

Other Examples

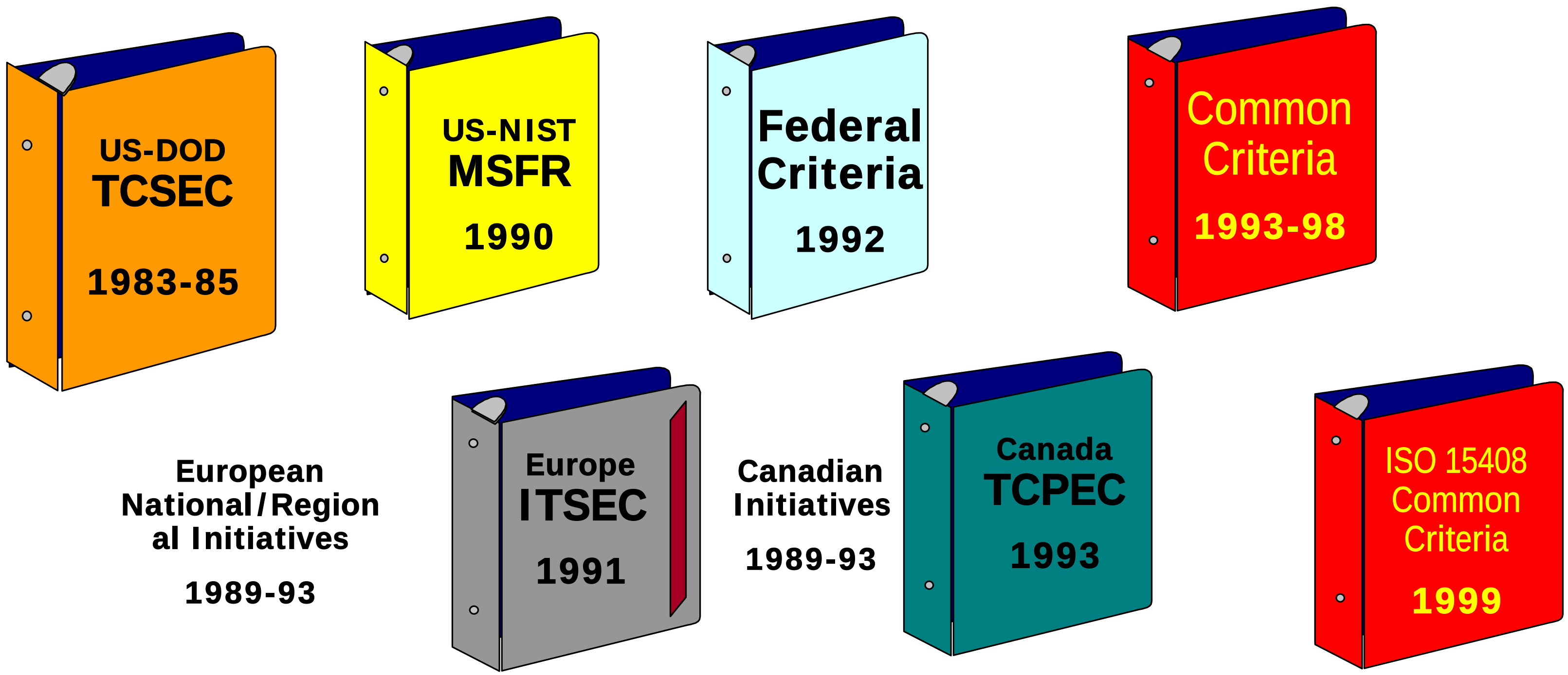
- Formal verification of protocols
  - Naval Research Laboratory Protocol Analyzer
    - For Crypto protocols
      - Key management (distribution)
      - Authentication protocols
- Verification of libraries
    - Entire system not verified
    - But components known okay
- High risk subsystems

Protocol Verification

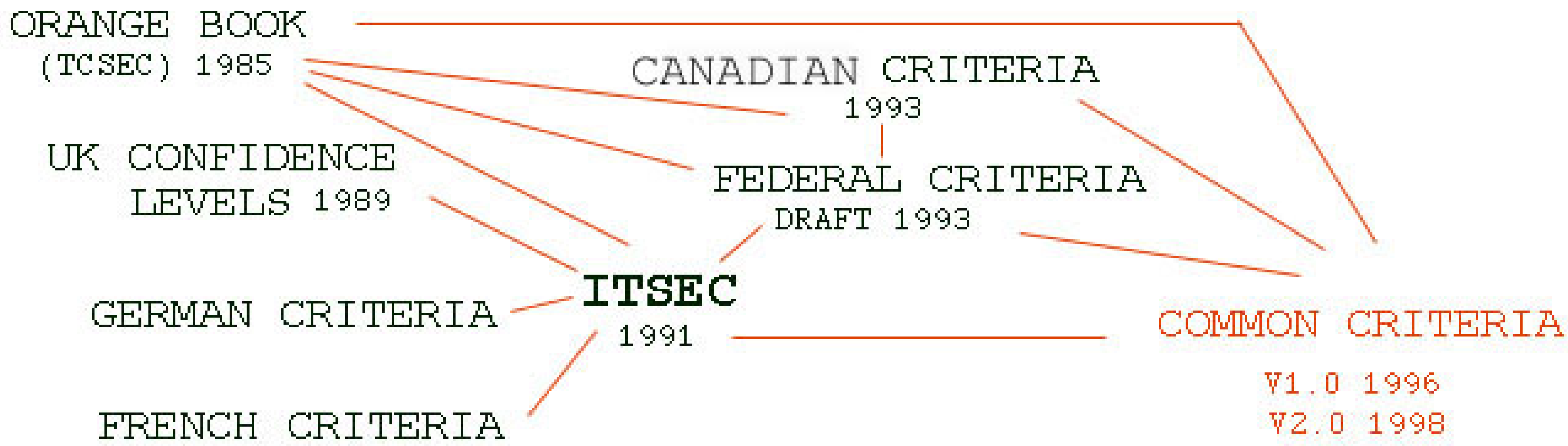
- Generating protocols that meet security specifications
  - BAN Logic
    - Believes, sees, once said
- Assumes cryptography secure
  - But cryptography is not enough

Common Criteria:
An Evolutionary Process

Decades of research and development…



Common Criteria: Origin



TCSEC

- Known as Orange Book, DoD 5200.28-STD
- Four trust rating divisions (classes)
  - D: Minimal protection
  - C (C1,C2): Discretionary protection
  - B (B1, B2, B3): Mandatory protection
  - A (A1): Highly-secure

TCSEC: The Original

- Trusted Computer System Evaluation Criteria
  - U.S. Government security evaluation criteria
  - Used for evaluating commercial products
- Policy model based on Bell-LaPadula
- Enforcement: Reference Validation Mechanism
  - Every reference checked by compact, analyzable body of code
- Emphasis on Confidentiality
- Metric: Seven trust levels:
  - D, C1, C2, B1, B2, B3, A1
  - D is "tried but failed"

TCSEC Class Assurances

- C1: Discretionary Protection
  - Identification
  - Authentication
  - Discretionary access control
- C2: Controlled Access Protection
  - Object reuse and auditing
- B1: Labeled security protection
  - Mandatory access control on limited set of objects
  - Informal model of the security policy

TCSEC Class Assurances *(continued)*

- B2: Structured Protections
  - Trusted path for login
  - Principle of Least Privilege
  - Formal model of Security Policy
  - Covert channel analysis
  - Configuration management
- B3: Security Domains
  - Full reference validation mechanism
  - Constraints on code development process
  - Documentation, testing requirements
- A1: Verified Protection
  - Formal methods for analysis, verification
  - Trusted distribution

How is Evaluation Done?

- Government-sponsored independent evaluators
  - Application: Determine if government cares
  - Preliminary Technical Review
    - Discussion of process, schedules
    - Development Process
    - Technical Content, Requirements
  - Evaluation Phase

TCSEC: Evaluation Phase

- Three phases
  - Design analysis
    - Review of design based on documentation
  - Test analysis
  - Final Review
- Trained independent evaluation
  - Results presented to Technical Review Board
  - Must approve before next phase starts
- Ratings Maintenance Program
  - Determines when updates trigger new evaluation

TCSEC: Problems

- Based heavily on confidentiality
  - Did not address integrity, availability
- Tied security and functionality
- Base TCSEC geared to operating systems
  - TNI: Trusted Network Interpretation
  - TDI: Trusted Database management System Interpretation

Later Standards

- CTCPEC – Canadian Trusted Computer Product Evaluation Criteria
- ITSEC – European Standard (Info Tech SEC)
  - Did not define criteria
  - Levels correspond to strength of evaluation
  - Includes code evaluation, development methodology requirements
  - Known vulnerability analysis
- CISR: Commercial outgrowth of TCSEC (Commercial International Security Requirements)
- FC: Modernization of TCSEC
- FIPS 140: Cryptographic module validation
- Common Criteria: International Standard
- SSE-CMM: Evaluates developer, not product

ITSEC: Levels

- E1: Security target defined, tested
  - Must have informal architecture description
- E2: Informal description of design
  - Configuration control, distribution control
- E3: Correspondence between code and security target
- E4: Formal model of security policy
  - Structured approach to design
  - Design level vulnerability analysis
- E5: Correspondence between design and code
  - Source code vulnerability analysis
- E6: Formal methods for architecture
  - Formal mapping of design to security policy
  - Mapping of executable to source code

ITSEC Problems:

- No validation that security requirements made sense
  - Product meets goals
  - But does this meet user expectations?
- Inconsistency in evaluations
  - Not as formally defined as TCSEC

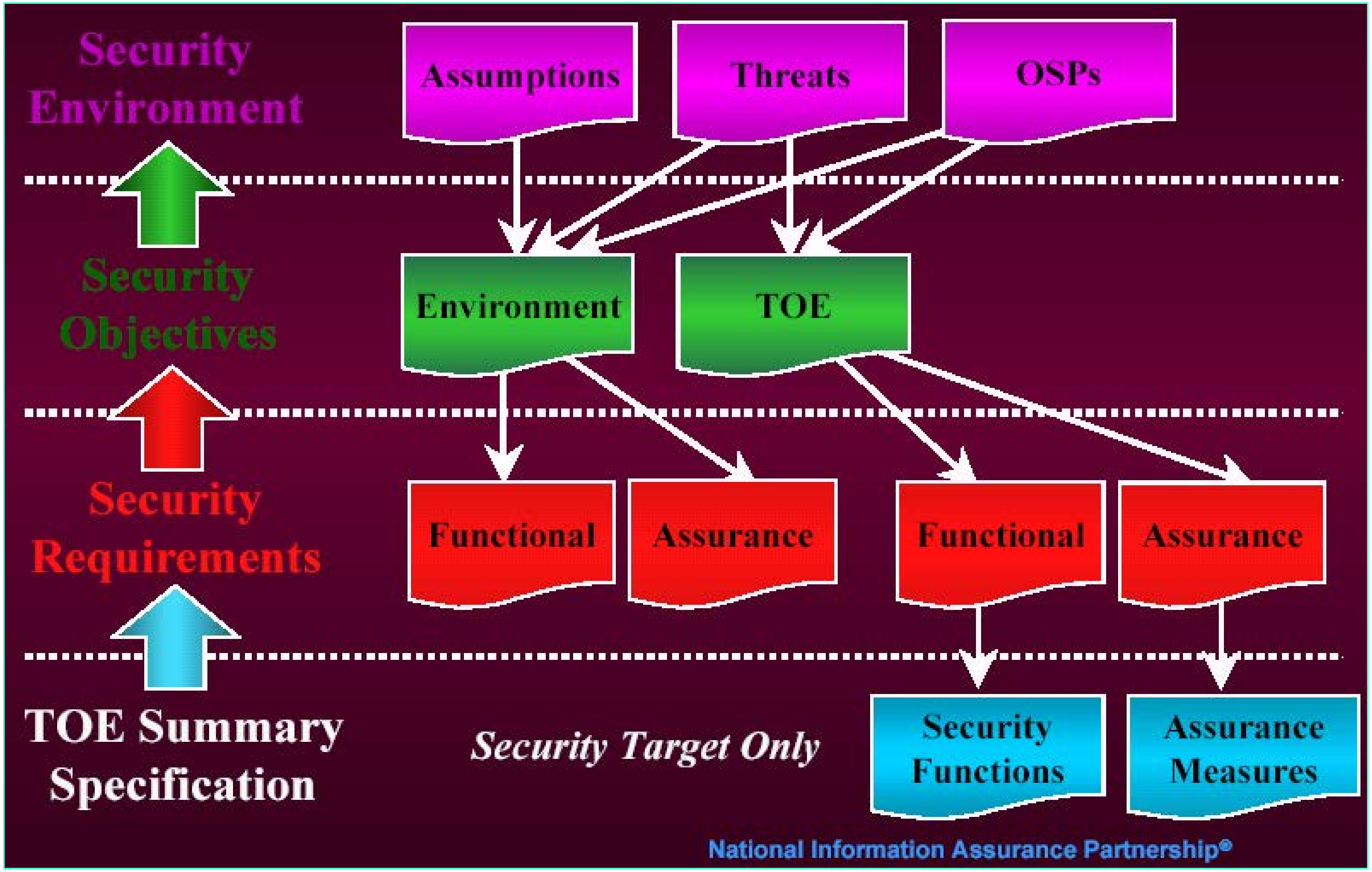
Common Criteria

- Replaced TCSEC, ITSEC
- 7 Evaluation Levels (functionally tested to formally designed and tested)
- Functional requirements, assurance requirements and evaluation methodology
- Functional and assurance requirements are organized hierarchically into: *class*, *family*, *component*, and, *element*. The components may have *dependencies*.

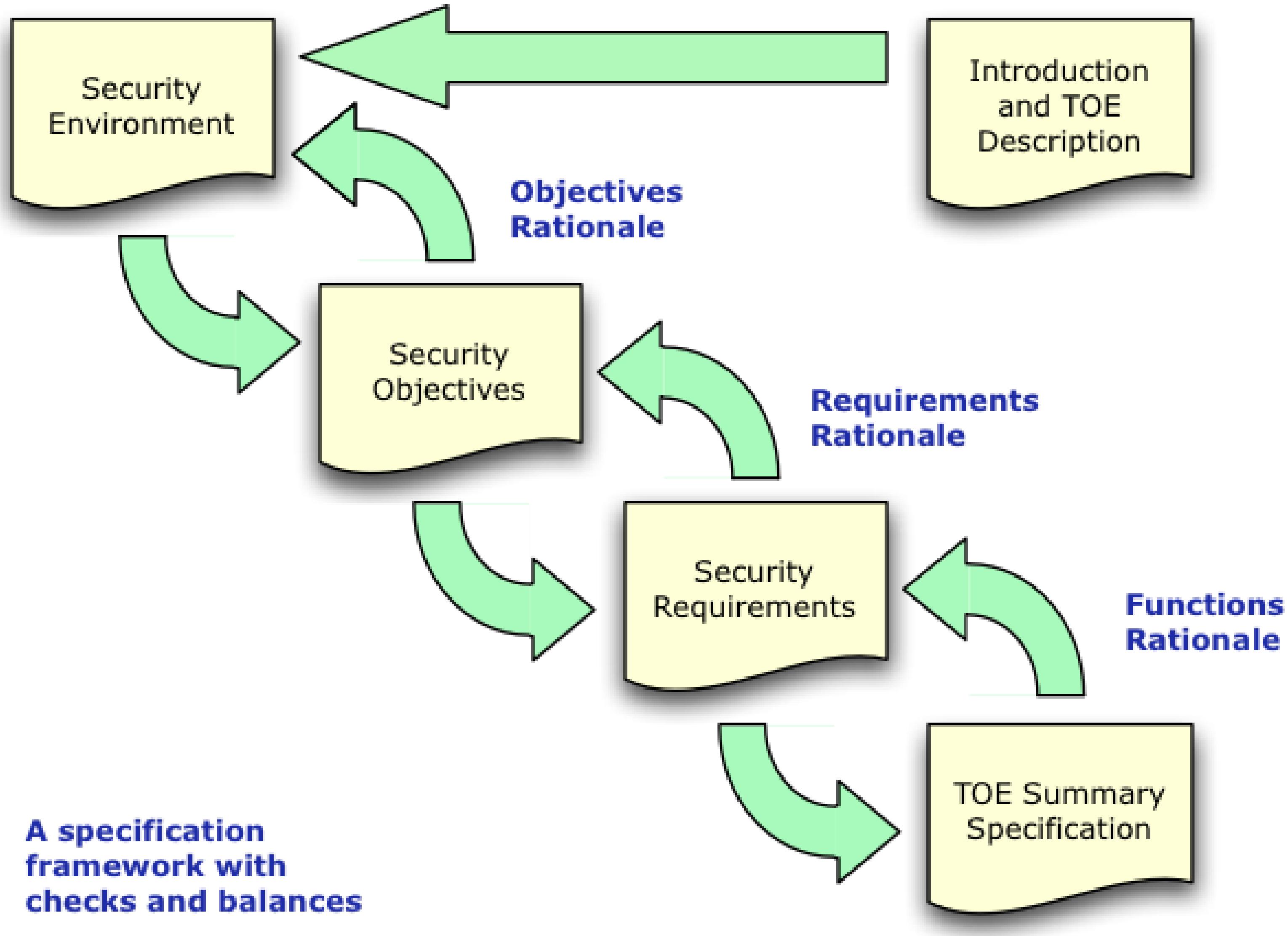
Key terms

- Protection profile
  - implementation-independent;
  - community/group; government sponsor, etc.
- Security Target
  - Set of security requirements that can be stated explicitly; product specific; implementation independent
- Target of Evaluation
  - Specific product

PP/ST Framework



Security Problem Definition



A specification framework with checks and balances

IT Security Requirements

CC defines two types of IT security requirements--

Functional Requirements

- for defining security behavior of the IT product or system:

• implemented requirements become security functions

Examples:

•*Identification & Authentication*
•*Audit*
•*User Data Protection*
•*Cryptographic Support*

Assurance Requirements

- for establishing confidence in security functions:

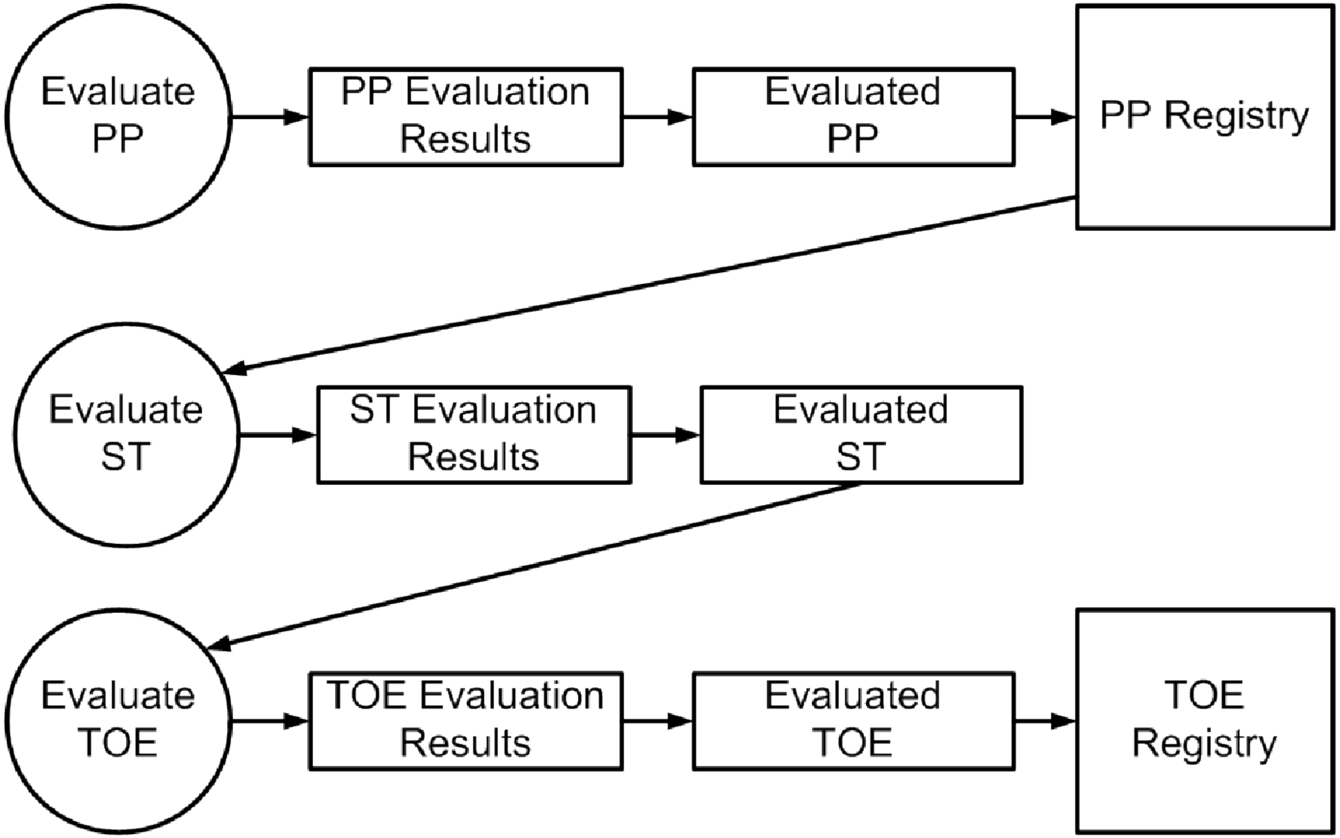
• correctness of implementation

• effectiveness in satisfying security objectives

Examples:

•*Development*
•*Configuration Management*
•*Life Cycle Support*
•*Testing*
•*Vulnerability Analysis*

Evaluation


Evaluate
PP
PP Evaluation
Results
Evaluated
PP
PP Registry
Evaluate
ST
ST Evaluation
Results
Evaluated
ST
Evaluate
TOE
TOE Evaluation
Results
Evaluated
TOE
TOE
Registry

Documentation

- Part 1: Introduction and General Model
  - https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5_marked_changes.pdf
- Part 2: Security Functional Requirements
  - https://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R5_marked_changes.pdf
- Part 3: Security Assurance Requirements
  - https://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R5_marked_changes.pdf
- CEM (Evaluation Methodology)
  - https://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R5_marked_changes.pdf
- Latest version: 3.1 Revision 5 (April 2017)
- https://www.commoncriteriaportal.org/cc/

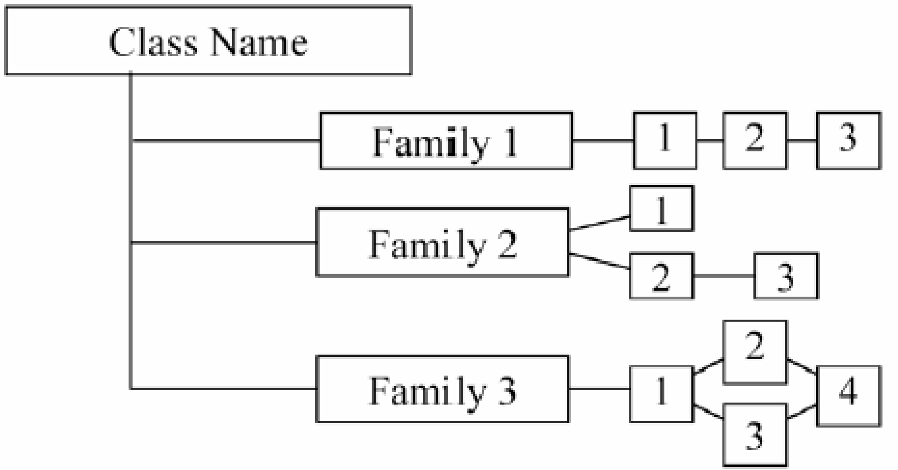
Class Decomposition

Class

Family

Components

Elements



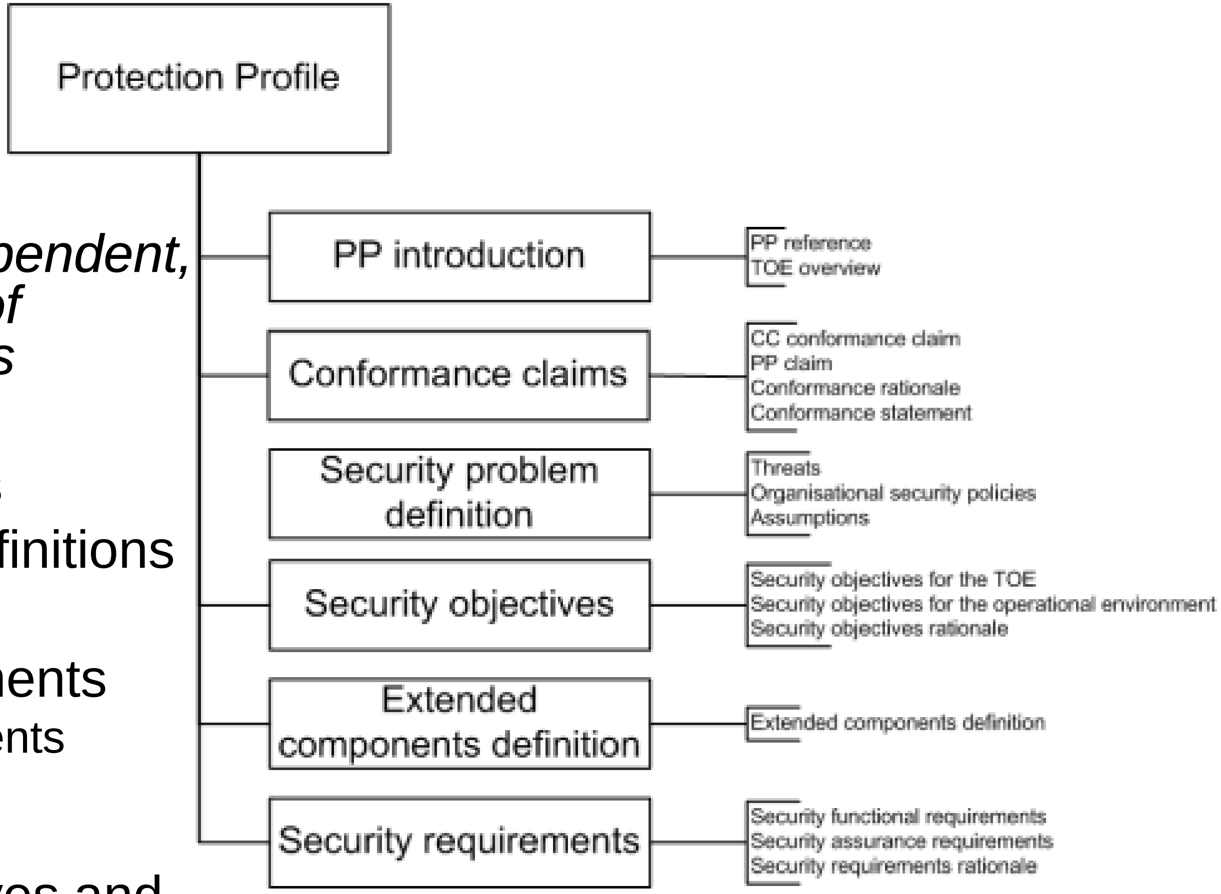
Note:

Applicable to both functional and assurance documents

CC Evaluation 1: Protection Profile

Implementation independent, domain-specific set of security requirements

- Narrative Overview
- Conformance Claims
- Security Problem Definitions
- Security Objectives:
- IT Security Requirements
  - Functional requirements drawn from CC set
  - Assurance level
- Rationale for objectives and requirements



CC Evaluation 2: Security Target

Specific requirements used to evaluate system

- Narrative introduction
- Conformance claims
- Security Problem Definition
- Security Objectives
  - How met
- Security Requirements
  - Environment and system
  - Drawn from CC set

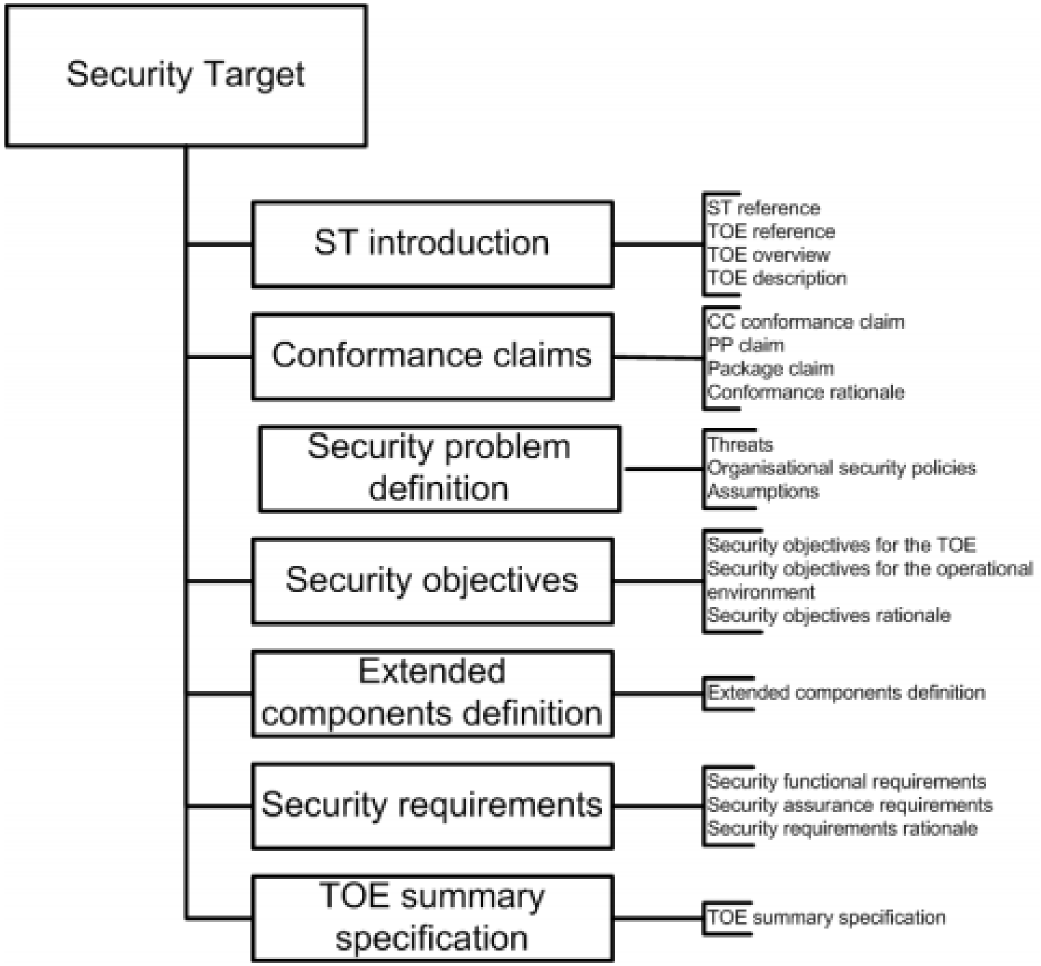


Figure 5 - Security Target contents

Common Criteria: Functional Requirements

- 323 page document
- 11 Classes
  - Security Audit, Communication, Cryptography, User data protection, ID/authentication, Security Management, Privacy, Protection of Security Functions, Resource Utilization, Access, Trusted paths
- Several families per class
- Lattice of components in a family

Common Criteria: Functional Requirements

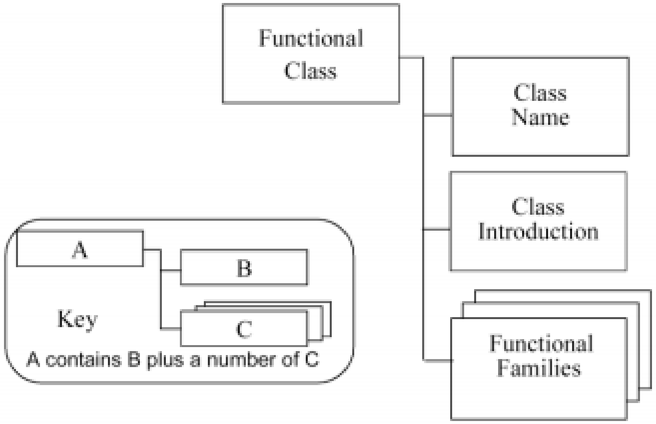


Figure 3 - Functional class structure

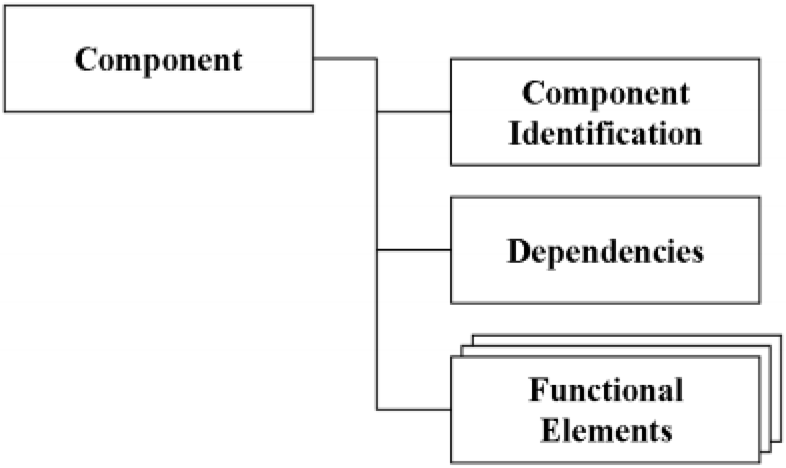


Figure 5 - Functional component structure

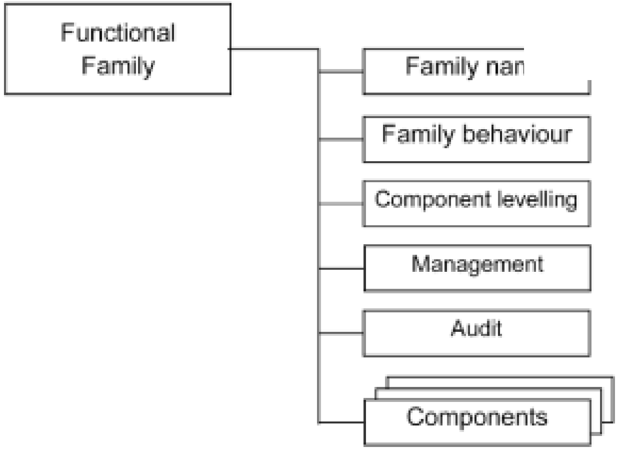
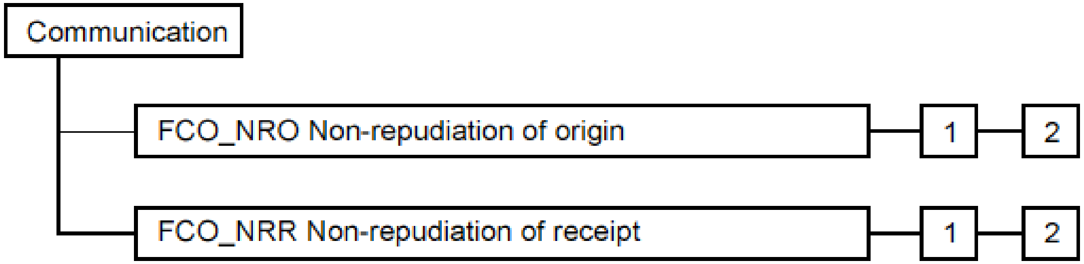


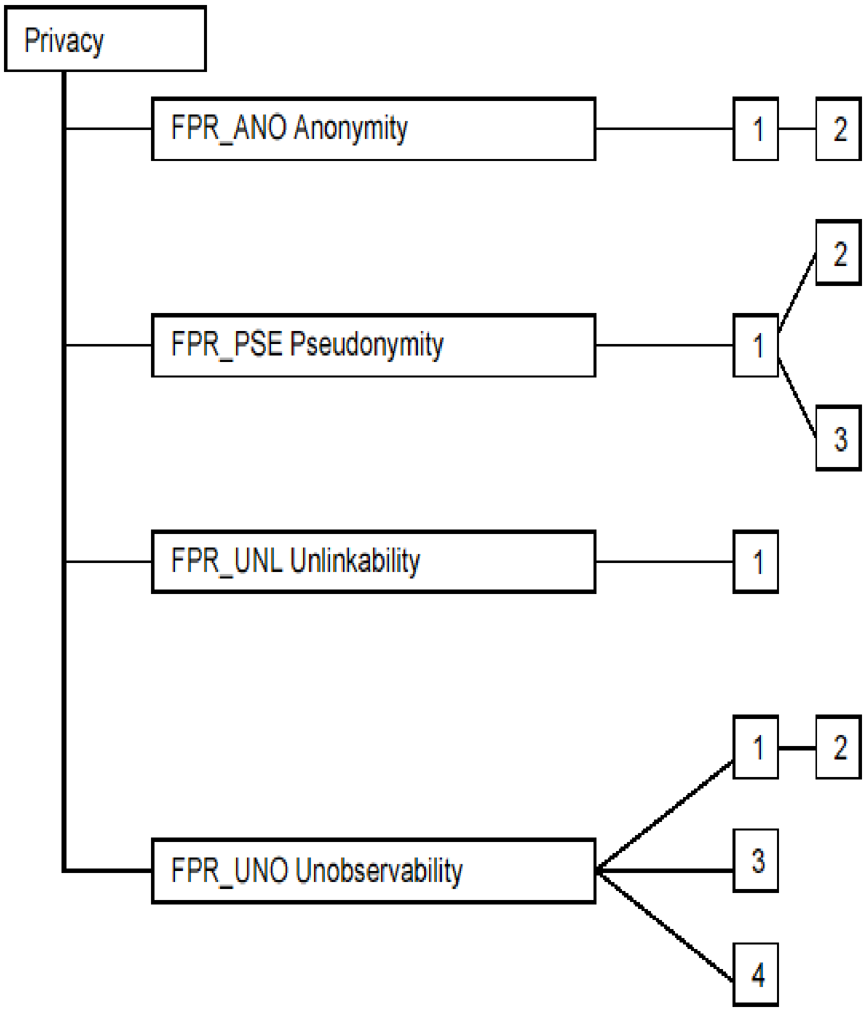
Figure 4 - Functional family structure

Class Example: Communication



- Non-repudiation of origin
  1. Selective Proof. Capability to request verification of origin
  2. Enforced Proof. All communication includes verifiable origin

Class Example: Privacy



1. Pseudonymity
   - The TSF shall ensure that [assignment: *set of users and/or subjects*] are unable to determine the real user name bound to [assignment: *list of subjects and/or operations and/or objects*]
   - The TSF shall be able to provide [assignment: *number of aliases*] aliases of the real user name to [assignment: *list of subjects*]
   - The TSF shall [selection: *determine an alias for a user, accept the alias from the user*] and verify that it conforms to the [assignment: *alias metric*]
2. Reversible Pseudonimity
   - …
3. Alias Pseudonimity
   1. …

Common Criteria: Assurance Requirements

- 247 page document
- 10 Classes
  - Protection Profile Evaluation, Security Target Evaluation, Configuration management, Delivery and operation, Development, Guidance, Life cycle, Tests, Vulnerability assessment, Maintenance
- Several families per class
- Lattice of components in family

Common Criteria: Evaluation Assurance Levels

1. Functionally tested
2. Structurally tested
3. Methodically tested and checked
4. Methodically designed, tested, and reviewed
5. Semi-formally designed and tested
6. Semi-formally verified design and tested
7. Formally verified design and tested

Common Criteria: Evaluation Process

- National Authority authorizes evaluators
  - U.S.: NIST accredits commercial organizations
  - Fee charged for evaluation
- Team of four to six evaluators
  - Develop work plan and clear with NIST
  - Evaluate Protection Profile first
  - If successful, can evaluate Security Target

Defining Requirements

ISO/IEC Standard 15408

Common Criteria

A flexible, robust catalogue of standardized IT security requirements (features and assurances)

Protection Profiles

Access Control
Identification
Authentication
Audit
Cryptography

- ✓ Operating Systems
- ✓ Database Systems
- ✓ Firewalls
- ✓ Smart Cards
- ✓ Applications
- ✓ Biometrics
- ✓ Routers
- ✓ VPNs

Consumer-driven security requirements in specific information technology areas

Industry Responds

Protection Profile

Firewall Security Requirements

Consumer statement of IT security requirements to industry in a specific information technology area

Security Targets

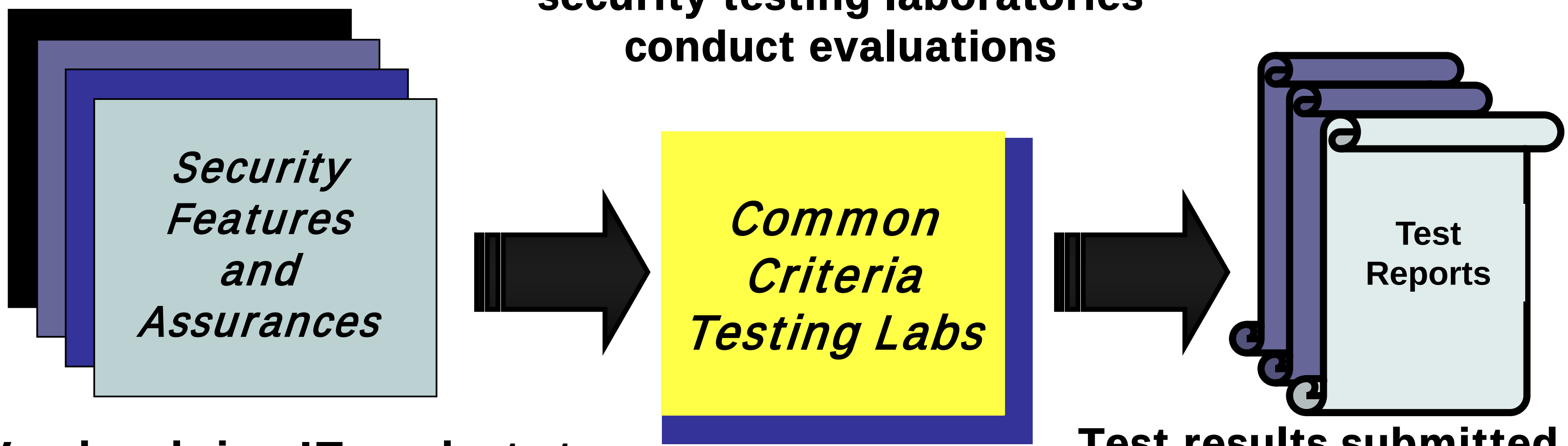
Security Features and Assurances

- ✓ CISCO Firewall
- ✓ Lucent Firewall
- ✓ Checkpoint Firewall
- ✓ Network Assoc. FW

Vendor statements of security claims for their IT products

Demonstrating Conformance

Private sector, accredited security testing laboratories conduct evaluations

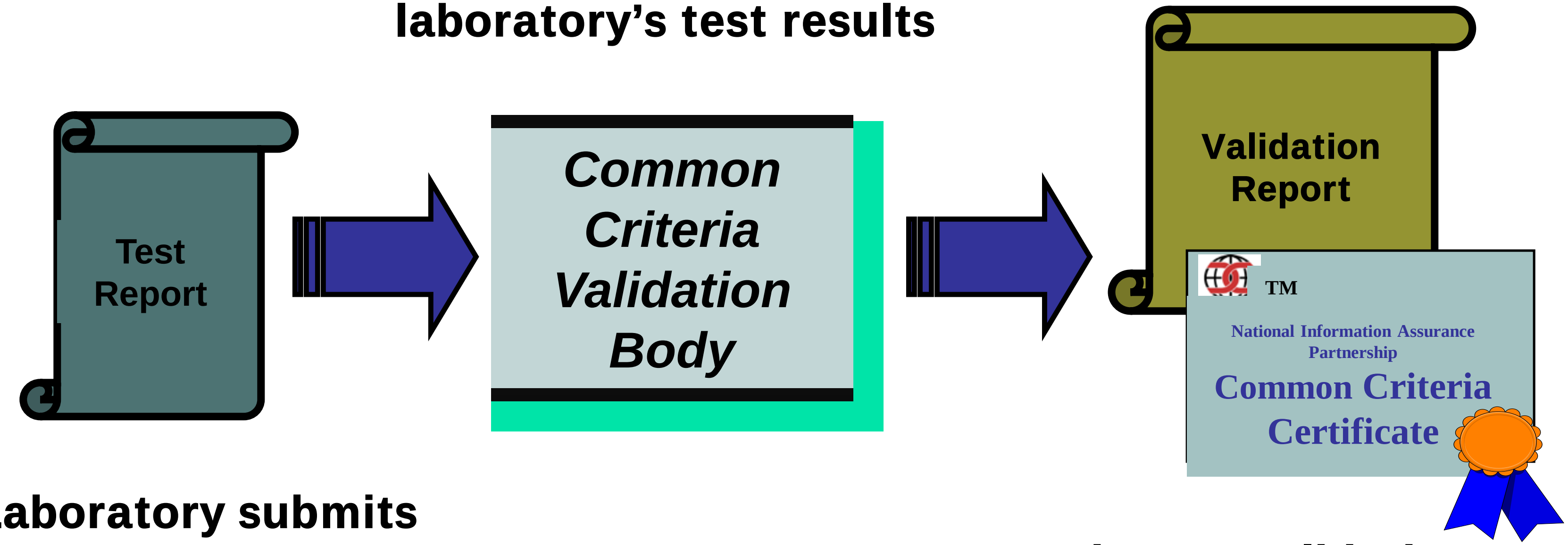


Vendors bring IT products to independent, impartial testing facilities for security evaluation

Test results submitted to the National Information Assurance Partnership (NIAP) for post-evaluation validation

Validating Test Results

Validation Body validates laboratory's test results



Laboratory submits test report to Validation Body

NIAP issues Validation Report and Common Criteria Certificate

Common Criteria: Statistics

2490 Certified Products by Category *		
Category	**Products**	**Archived**
Access Control Devices and Systems	69	60
Biometric Systems and Devices	3	0
Boundary Protection Devices and Systems	79	122
Data Protection	70	91
Databases	31	53
Detection Devices and Systems	12	57
ICs, Smart Cards and Smart Card-Related Devices and Systems	1190	32
Key Management Systems	22	28
Mobility	32	18
Multi-Function Devices	194	180
Network and Network-Related Devices and Systems	251	234
Operating Systems	104	74
Other Devices and Systems	294	314
Products for Digital Signatures	102	8
Trusted Computing	37	0
Totals:	**2490**	**1271**
Grand Total:		**3761**

** A Certified Product may have multiple Categories associated with it.*

Source: **https://www.commoncriteriaportal.org/products/stats/**

Common Criteria: Statistics

Certified Products by Assurance Level and Certification Date																					
EAL	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	Total
EAL1	0	0	0	0	0	0	1	1	6	3	1	0	1	10	2	2	3	3	8	6	47
EAL1+	1	0	0	0	0	0	0	0	17	0	2	11	2	0	1	2	1	0	0	1	38
EAL2	0	0	0	0	0	0	1	0	8	1	7	2	3	1	10	12	18	15	23	9	110
EAL2+	0	0	0	1	1	1	2	2	8	8	8	4	5	10	11	27	59	76	66	36	325
EAL3	0	0	0	0	0	0	0	0	10	3	1	9	5	1	7	12	9	2	3	2	64
EAL3+	0	0	0	0	0	2	1	1	37	10	12	11	12	19	7	23	17	19	10	7	188
EAL4	0	1	0	1	0	0	0	0	28	5	9	4	6	2	7	2	0	5	2	8	80
EAL4+	0	1	1	2	2	3	3	2	142	58	66	56	60	87	62	51	57	56	52	33	794
EAL5	0	0	0	0	0	0	0	0	6	3	2	0	1	0	0	0	0	3	1	3	19
EAL5+	0	0	0	0	0	0	3	0	50	27	31	43	35	27	56	51	43	69	68	45	548
EAL6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EAL6+	0	0	0	0	0	0	0	0	0	0	2	3	0	4	5	6	10	8	12	20	70
EAL7	0	0	0	0	0	0	0	0	0	0	1	0	0	0	4	0	0	0	0	0	5
EAL7+	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	2
Basic	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Medium	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
US Standard	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
None	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	8	13	23	78	74	200
Totals:	**1**	**2**	**1**	**4**	**3**	**6**	**11**	**6**	**312**	**118**	**142**	**144**	**130**	**161**	**176**	**196**	**230**	**279**	**323**	**245**	**2490**

Source: https://www.commoncriteriaportal.org/products/stats/

Common Criteria: Statistics

Protection Profiles

Statistics | Download CSV | Collaborative Protection Profiles | Archived Protection Profiles

expand/collapse all categories

- ⊞ Access Control Devices and Systems – 4 Protection Profiles
- ⊞ Biometric Systems and Devices – 2 Protection Profiles
- ⊞ Boundary Protection Devices and Systems – 11 Protection Profiles
- ⊞ Data Protection – 10 Protection Profiles
- ⊞ Databases – 3 Protection Profiles
- ⊞ ICs, Smart Cards and Smart Card-Related Devices and Systems – 75 Protection Profiles
- ⊞ Key Management Systems – 4 Protection Profiles
- ⊞ Mobility – 4 Protection Profiles
- ⊞ Multi-Function Devices – 2 Protection Profiles
- ⊞ Network and Network-Related Devices and Systems – 12 Protection Profiles
- ⊞ Operating Systems – 2 Protection Profiles
- ⊞ Other Devices and Systems – 49 Protection Profiles
- ⊞ Products for Digital Signatures – 19 Protection Profiles
- ⊞ Trusted Computing – 6 Protection Profiles

Source: https://www.commoncriteriaportal.org/pps/

Certified products: https://www.commoncriteriaportal.org/products/

Summary

- Assurance
  - Formal methods
  - Common Criteria