Tivoli software

business software

Security Patterns Workshop

Bob Blakley
Chief Scientist, Security and Privacy
IBM Tivoli Software

15 Oct 2002

IBM Software Group

Tivoli software

e business software

Design Patterns: Background

IBM Software Group

Why Patterns?

- History:
  - ➢ Long history of TOG security specifications
  - ➢ structural guidelines
  - ➢ C language interface definitions
- Problems
  - ➢ most systems already exist
    - – can't easily be revised to conform
  - ➢ C language is interface definitions decreasingly relevant
  - ➢ Audience needs guidance, not prescription
    - – adaptable to its own problems, dealing with existing systems
- Need
  - ➢ Language independent security guidance
  - ➢ Instructional rather than prescriptive

Who is The Audience for Patterns?

- A system designer or architect
- With a specific security problem in a specifc context
- Who would like to know how the Open Group's security experts would approach[1] his (or her) problem
- But doesn't want to come to our meetings or pay our consulting rates

[1]Not solve!

What is a Pattern?

- “A pattern is a named nugget of instructive information that captures the essential structure and insight of a successful family of proven solutions to a recurring problem that arises within a certain context and system of forces.”

What Does a Pattern Look Like?

- Minimally**:**
  - **Pattern name.** A memorable and descriptive way to refer to the pattern.
  - **The problem.** A description of the contexts and situations in which the pattern is useful.
  - **The solution.** A specific but flexible approach to solving the problem.
  - **Consequences.** Implementing the solution described in the pattern will require making specific tradeoffs among competing forces. These tradeoffs and their consequences are described.
- Specifically:
  - We use (approximately) GOF format

How Do You Design A System Using Patterns?

- Stepwise Refinement
  - Start with a high-level solution and refine it until you have the level of detail you require to build the system
- Generative Sequences
  - Sequence of operations
  - No backtracking
  - Well-identified choices at each step
  - Each step involves a choice of which patterns to apply

Tivoli software

IBM

e business software

The Open Group Security Design Patterns

IBM Software Group

Open Group Security Pattern Catalogs

- Protected System Catalog
  - Contains structural design patterns which facilitate construction of systems which protect valuable resources against unauthorized disclosure or modification.
- Available System Catalog
  - Contains structural design patterns which facilitate construction of systems which provide predictable uninterrupted access to the services and resources they offer to users.

Open Group Available System Pattern Catalog

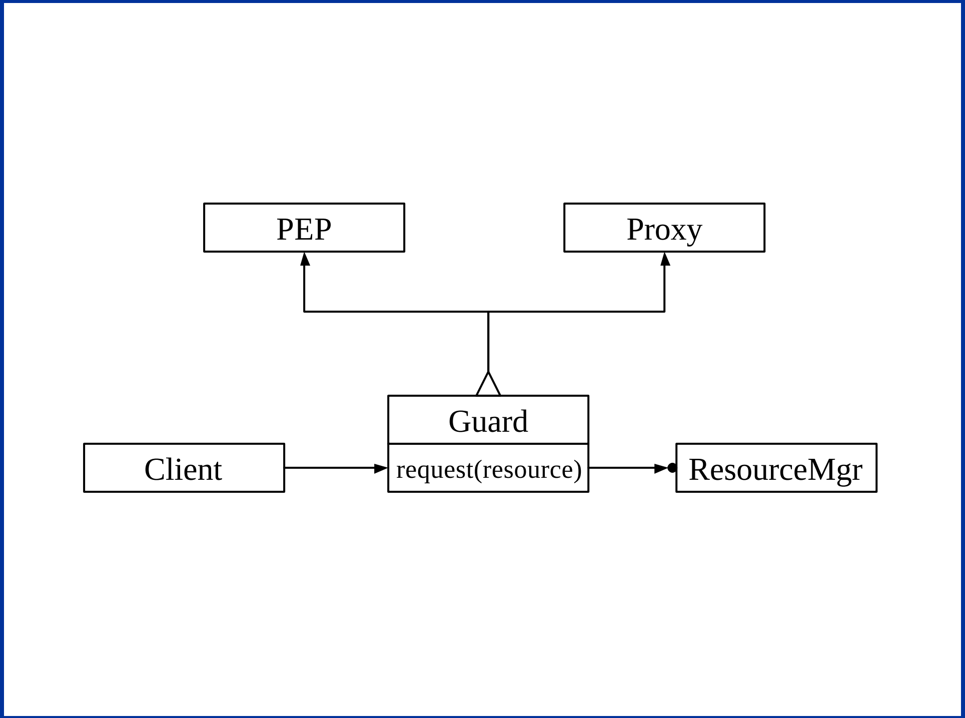
- Available System Patterns
  - Recoverable Component
  - Checkpointed System
  - Cold Standby
  - Comparator-Checked Fault-Tolerant System
  - Journalled Component
  - Hot Standby
  - External Storage
  - Replicated System
  - Error Detection/Correction

Open Group Protected System Pattern Catalog

- Protected System Patterns
  - Protected System
  - Policy Enforcement Point
  - Subject Descriptor
  - Secure Communication
  - Security Context
  - Secure Association
  - Policy
  - Proxy Patterns
    - Trusted Proxy
    - Authenticating Impersonator
    - Identity-Asserting Impersonator
    - Delegate
    - Authorizing Proxy
    - Login Tunnel

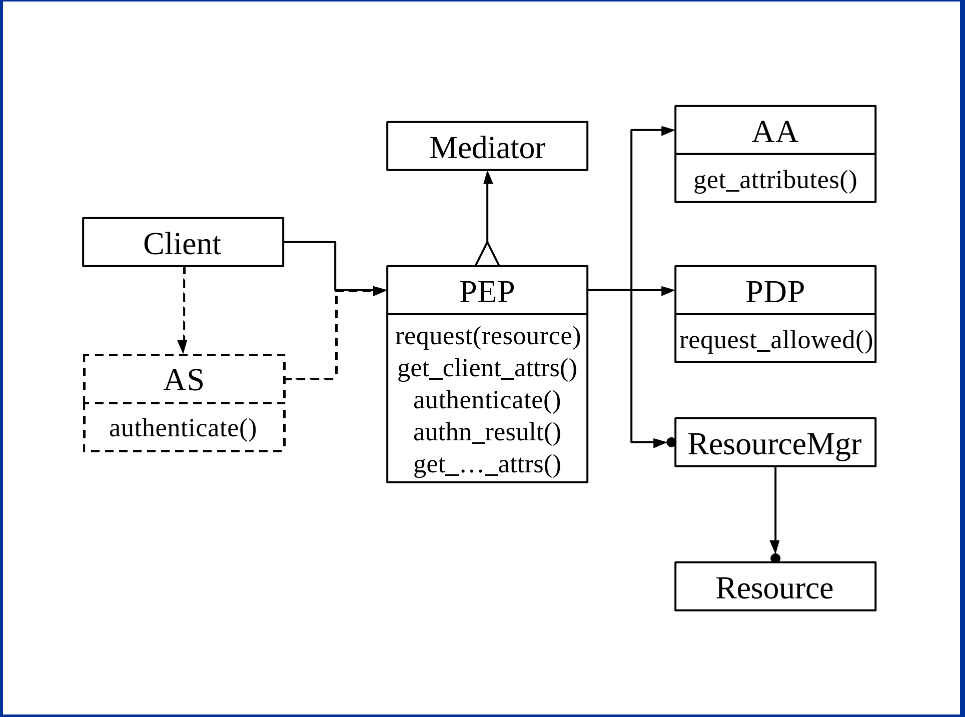
Protected System ***

Intent: Structure a system so that all access by clients to resources is mediated by a guard which enforces a security policy.



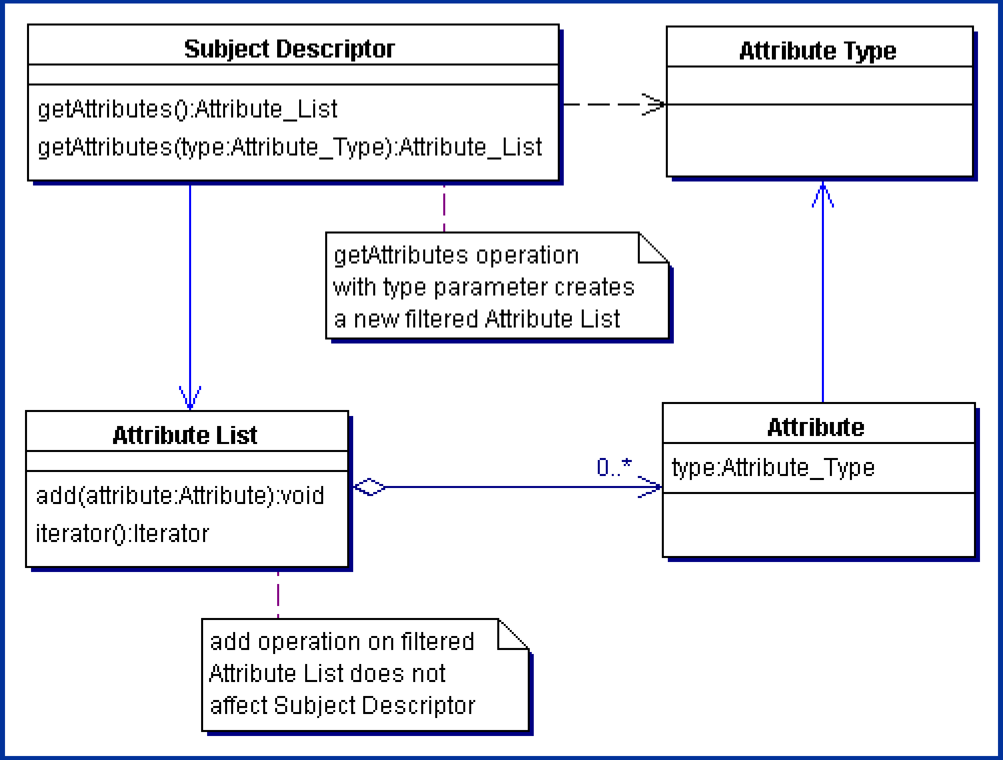
Policy Enforcement Point **

Intent: Isolate policy enforcement to a discrete component of an information system; ensure that policy enforcement activities are performed in the proper sequence.



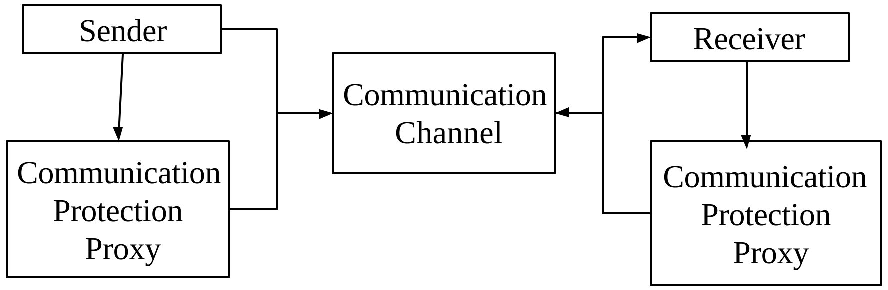
Subject Descriptor *

Intent: Provide access to security-relevant attributes of an entity on whose behalf operations are to be performed.


Subject Descriptor
getAttributes():Attribute_List
getAttributes(type:Attribute_Type):Attribute_List
Attribute Type
getAttributes operation with type parameter creates a new filtered Attribute List
Attribute List
add(attribute:Attribute):void
iterator():Iterator
0..*
Attribute
type:Attribute_Type
add operation on filtered Attribute List does not affect Subject Descriptor

Secure Communication ***

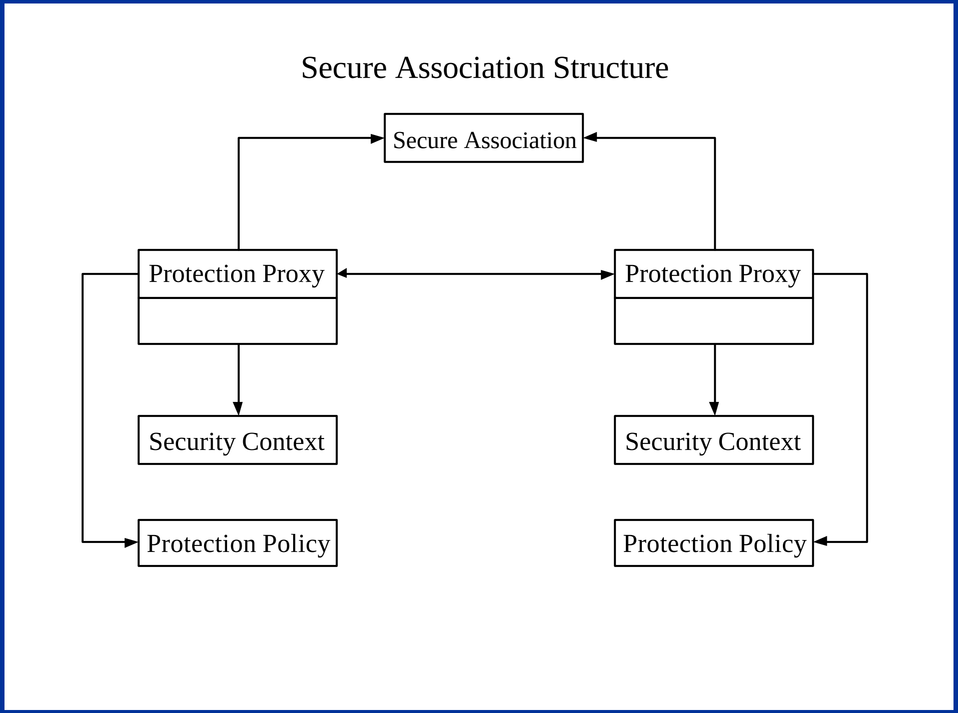
Intent: Ensure that mutual security policy objectives are met when there is a need for two parties to communicate in the presence of threats.



Sender
Communication Protection Proxy
Communication Channel
Receiver
Communication Protection Proxy

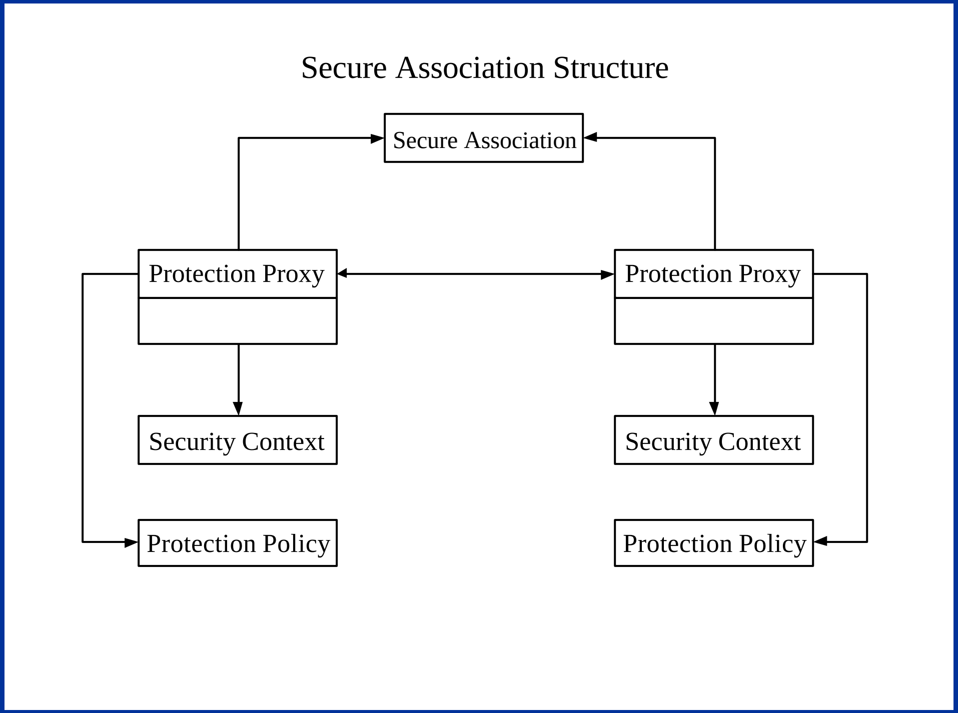
Security Context ***

Intent: Provide a container for, and mediate access to, security attributes and data relating to a particular process, operation or action.



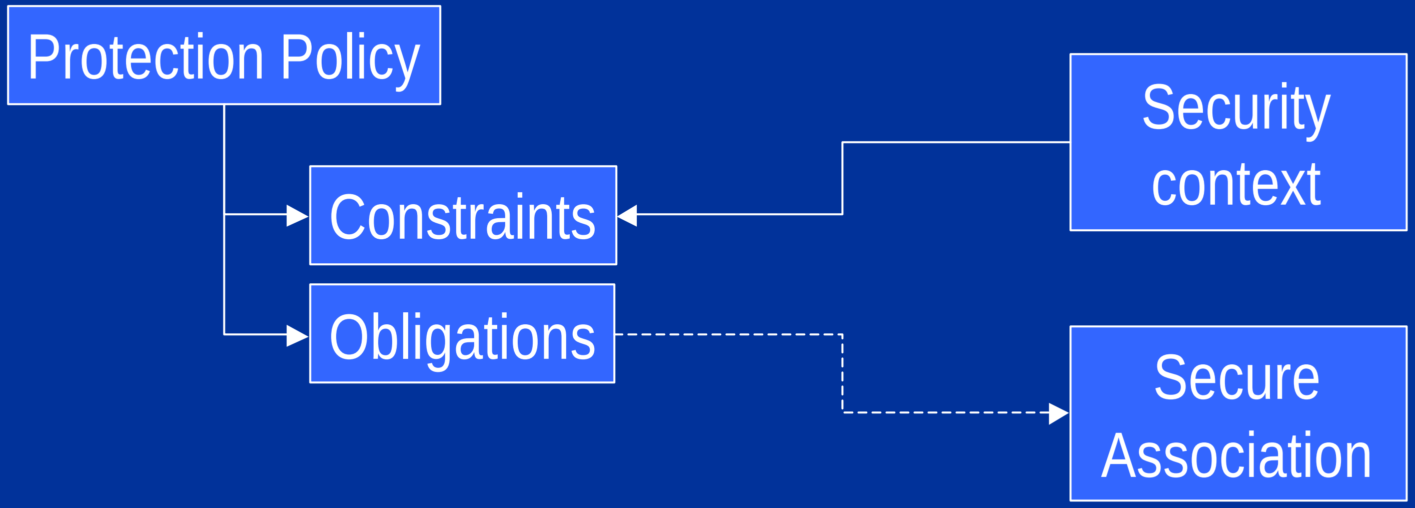
Secure Association ***

Intent: establish and maintain a security relationship, between two entities that wish to communicate securely, in line with mutual security policy objectives, across a communication link that is subject to a well-known set of communication related threats.



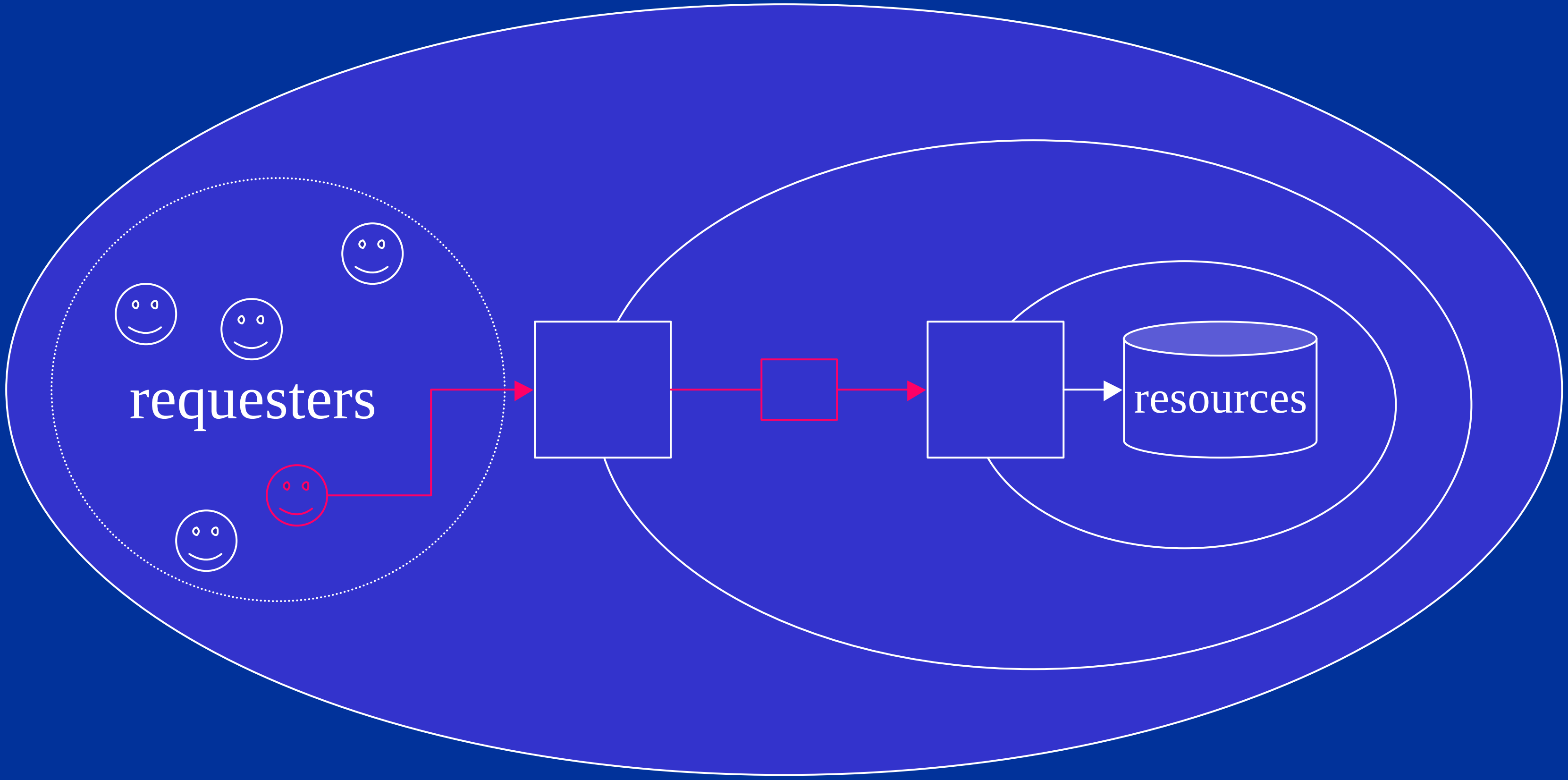
Policy ***

Intent: Define which operations are allowed to occur and which operations are required to occur in a protected system

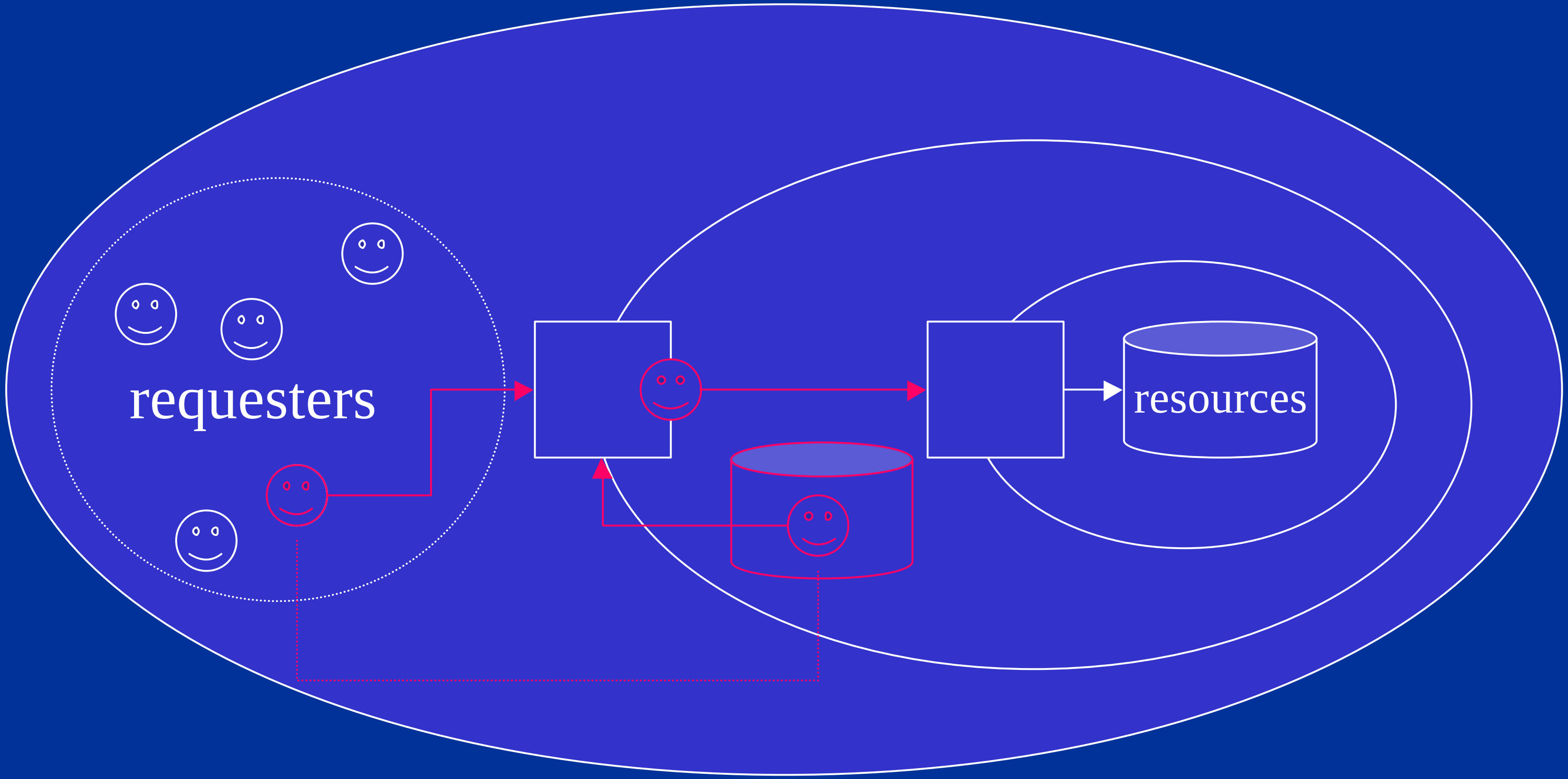


Constraints govern what must be in the security context in order for the policy to allow an operation
Obligations govern what secure associations may be created

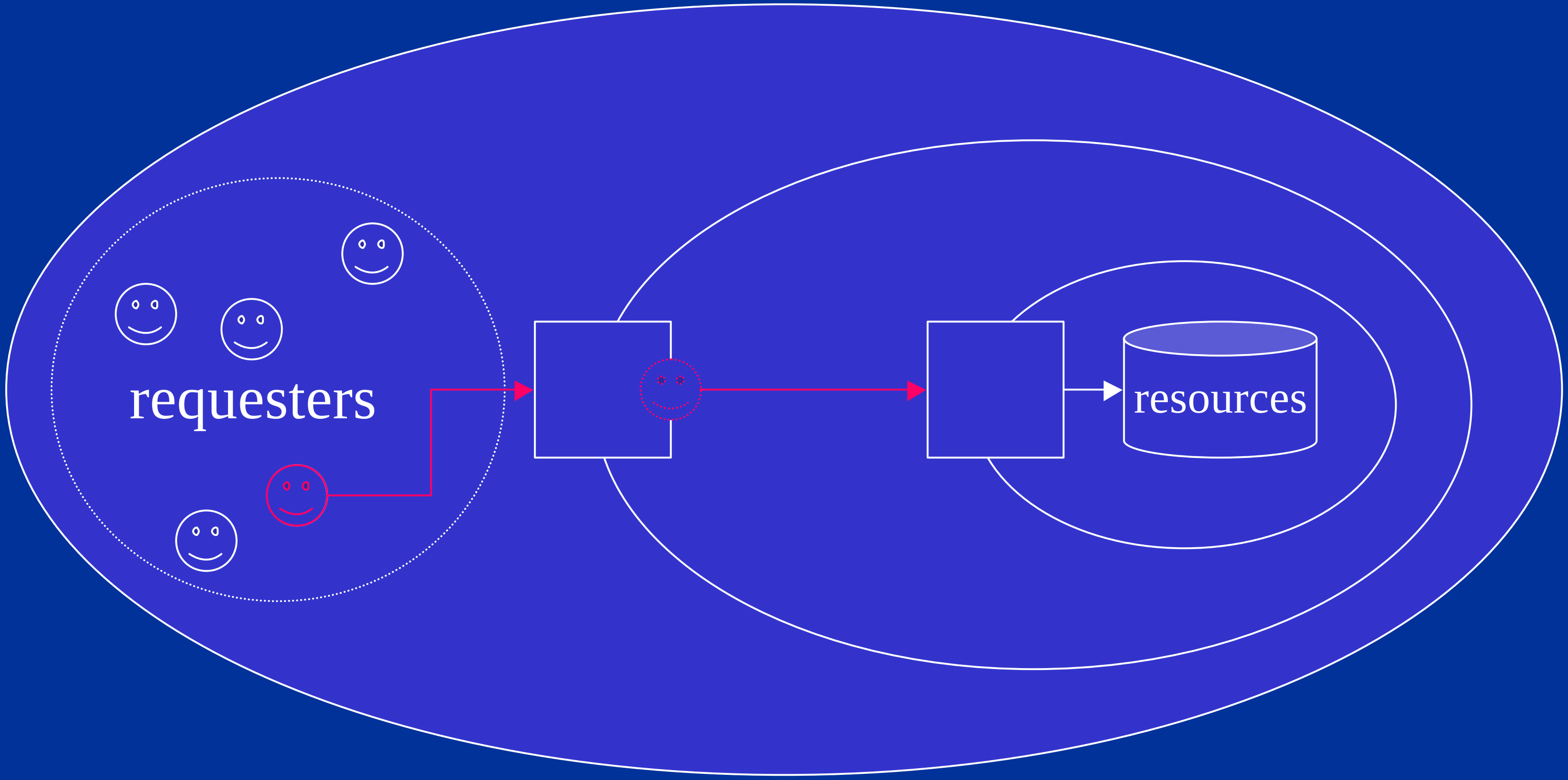
Trusted Proxy



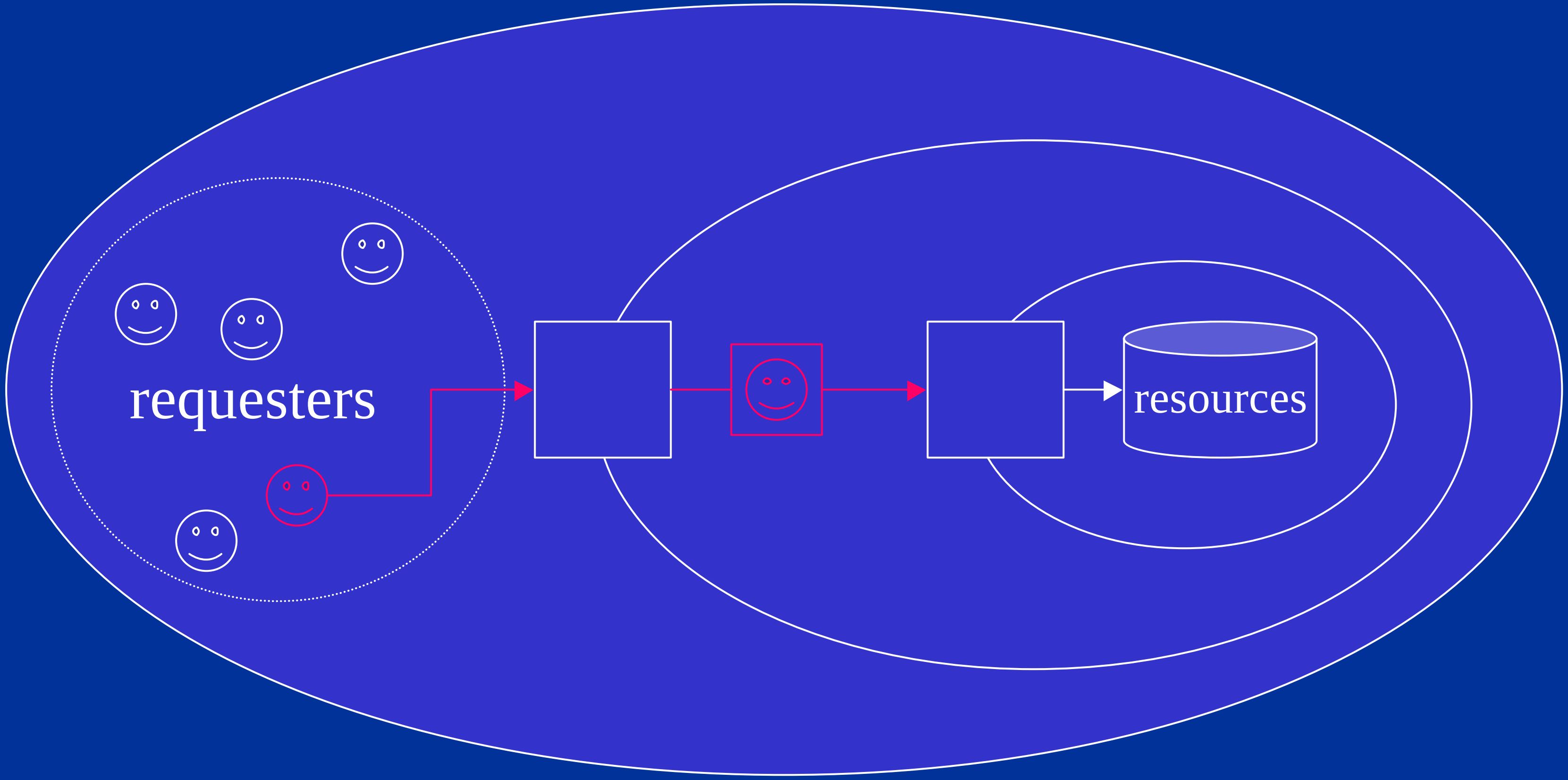
Authenticating Impersonator



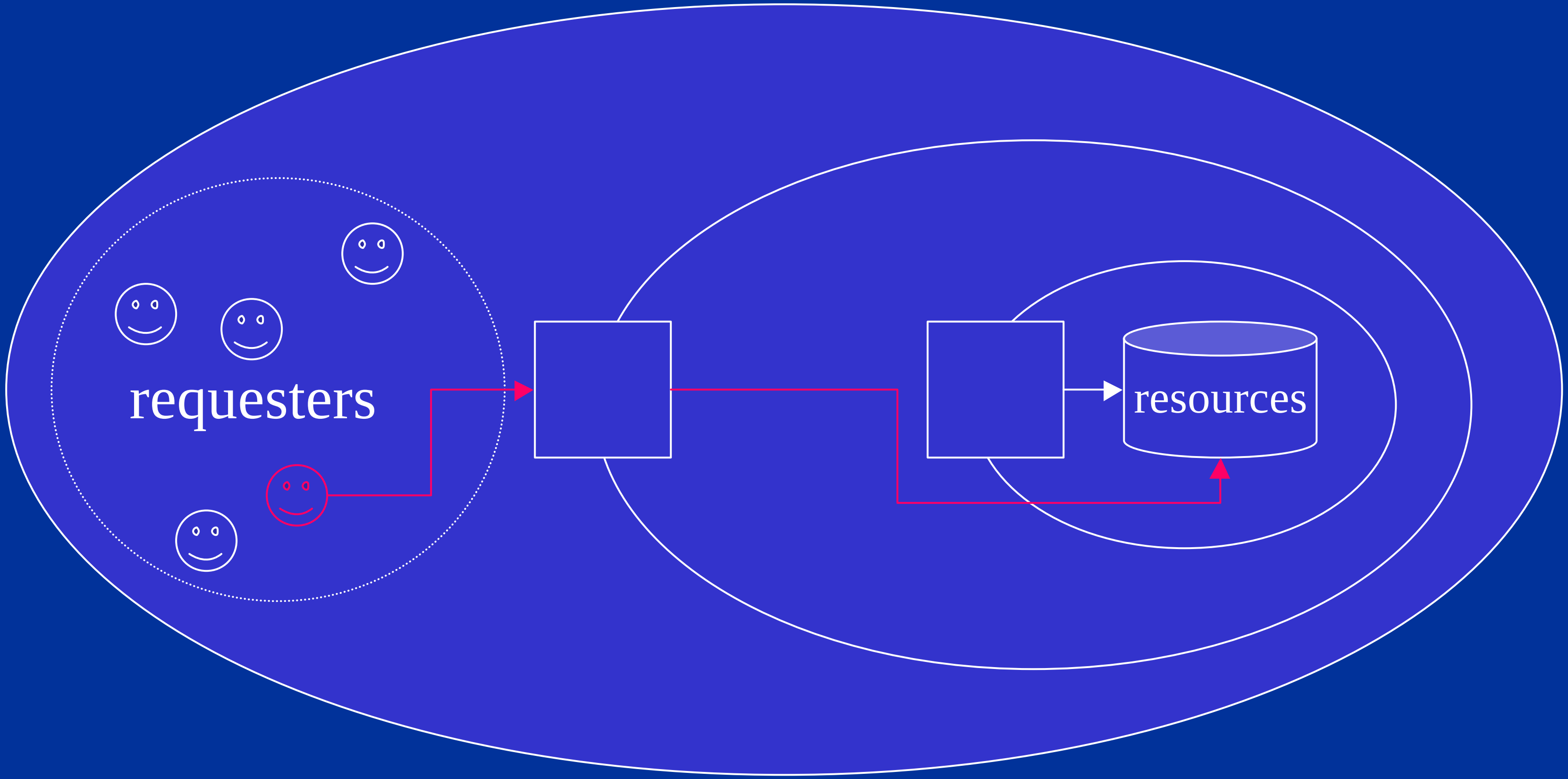
Identity-Asserting Impersonator



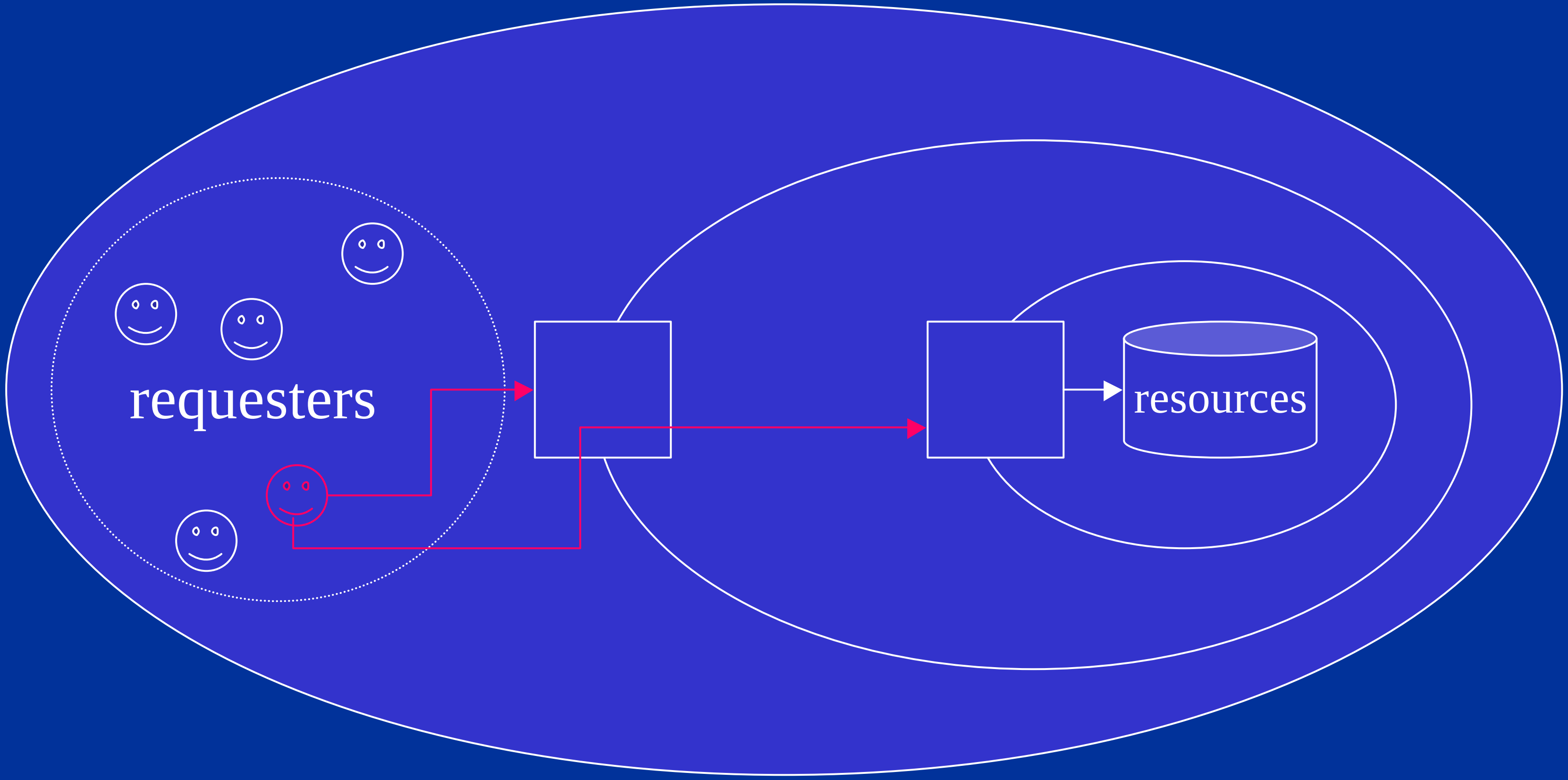
Delegate



Authorizing Proxy



Login Tunnel



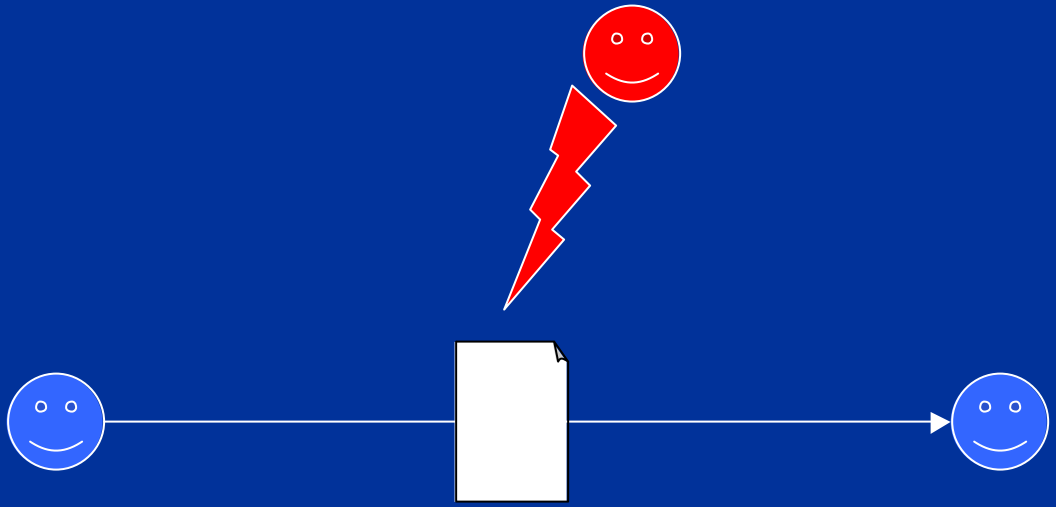
Open Group Protected System Generative Sequence

1. Identify resources and actors
2. Define one or more PROTECTED SYSTEM instances
3. For each PS, define POLICY
4. For each POLICY, define required SECURE COMMUNICATIONs
5. Derive TARGET DESCRIPTOR requirements from POLICY
6. Derive SECURITY CONTEXT from POLICY at each end of SC
7. Derive SECURE ASSOCIATION and SUBJECT DESCRIPTOR from SCTXTs for each SC
8. Examine each SC to determine whether it needs to be factored into a PS with multiple SCs

Note: we haven't updated this to accommodate Proxy patterns yet

Note: we haven't addressed deriving audit requirements from policy

Example Problem: Secure Email



Tivoli software

Tivoli software

IBM

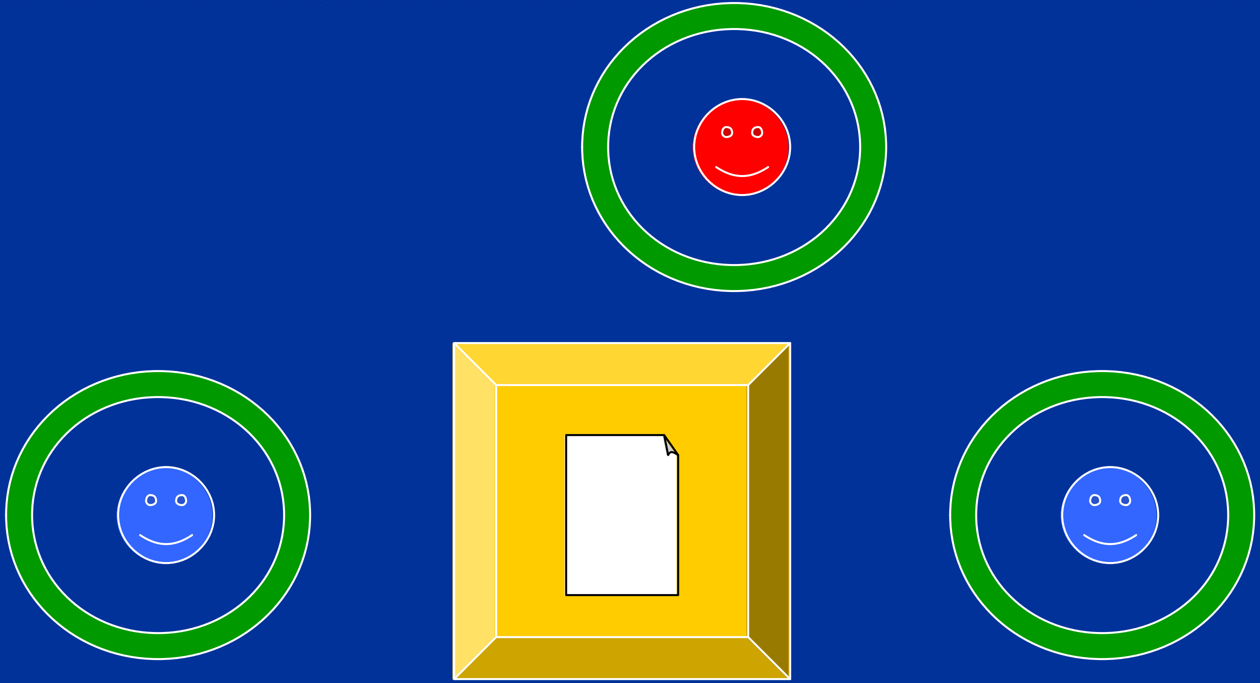
e business software

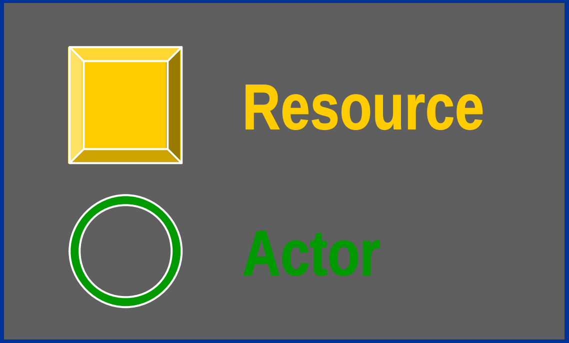
Using The Open Group Security Design Patterns

An Example

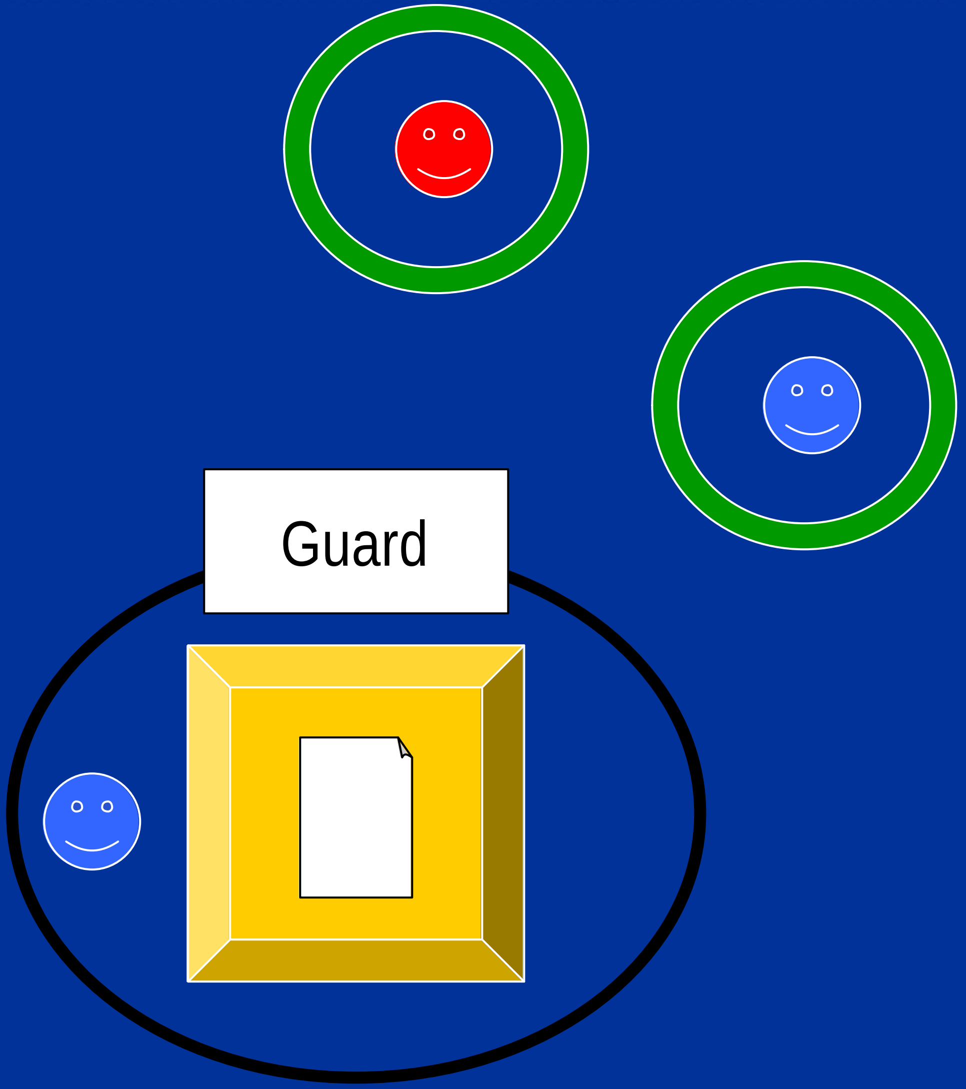
IBM Software Group

Refinement 1: Identify Resources and Actors

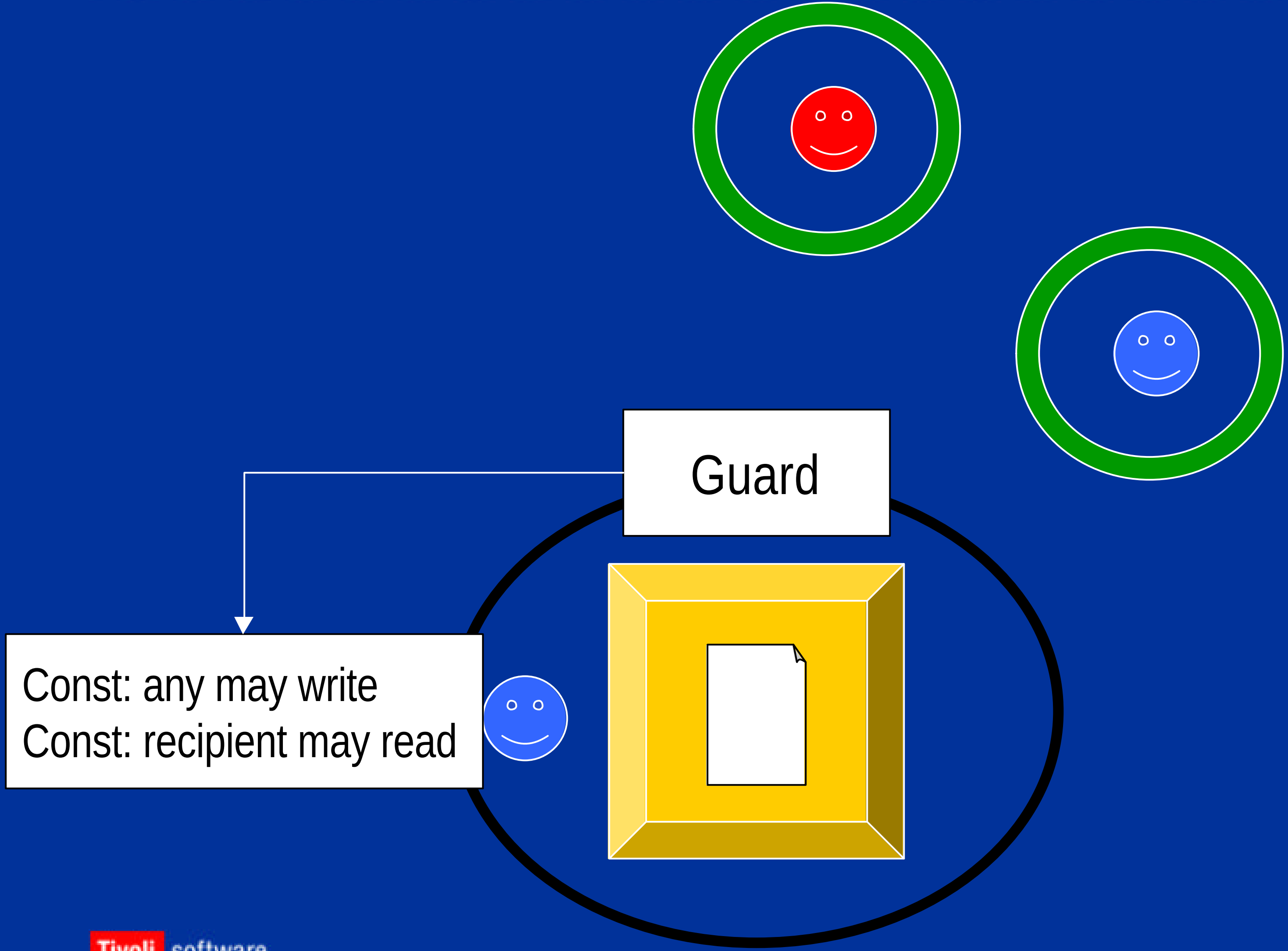




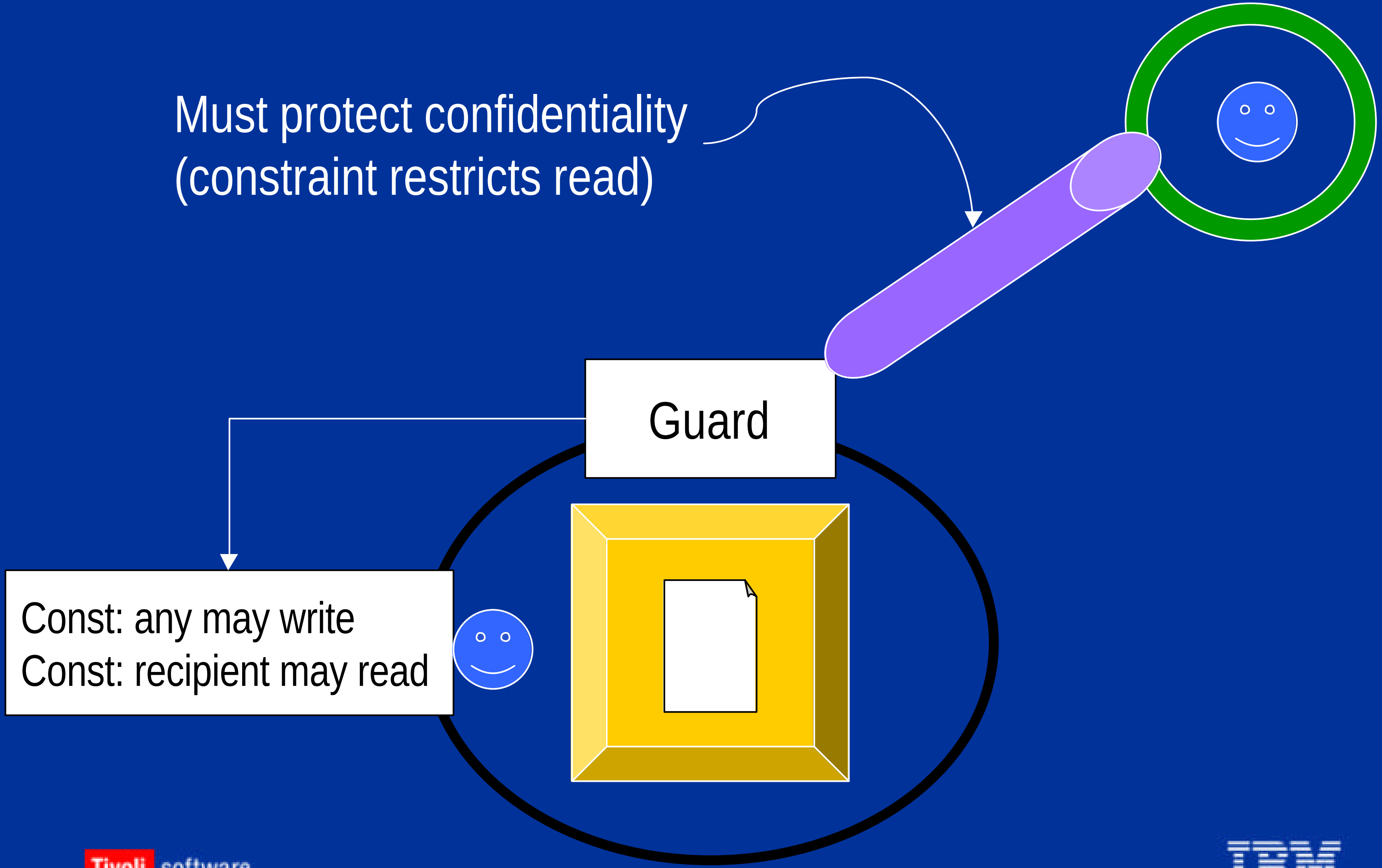
Refinement 2: Define PS Instances



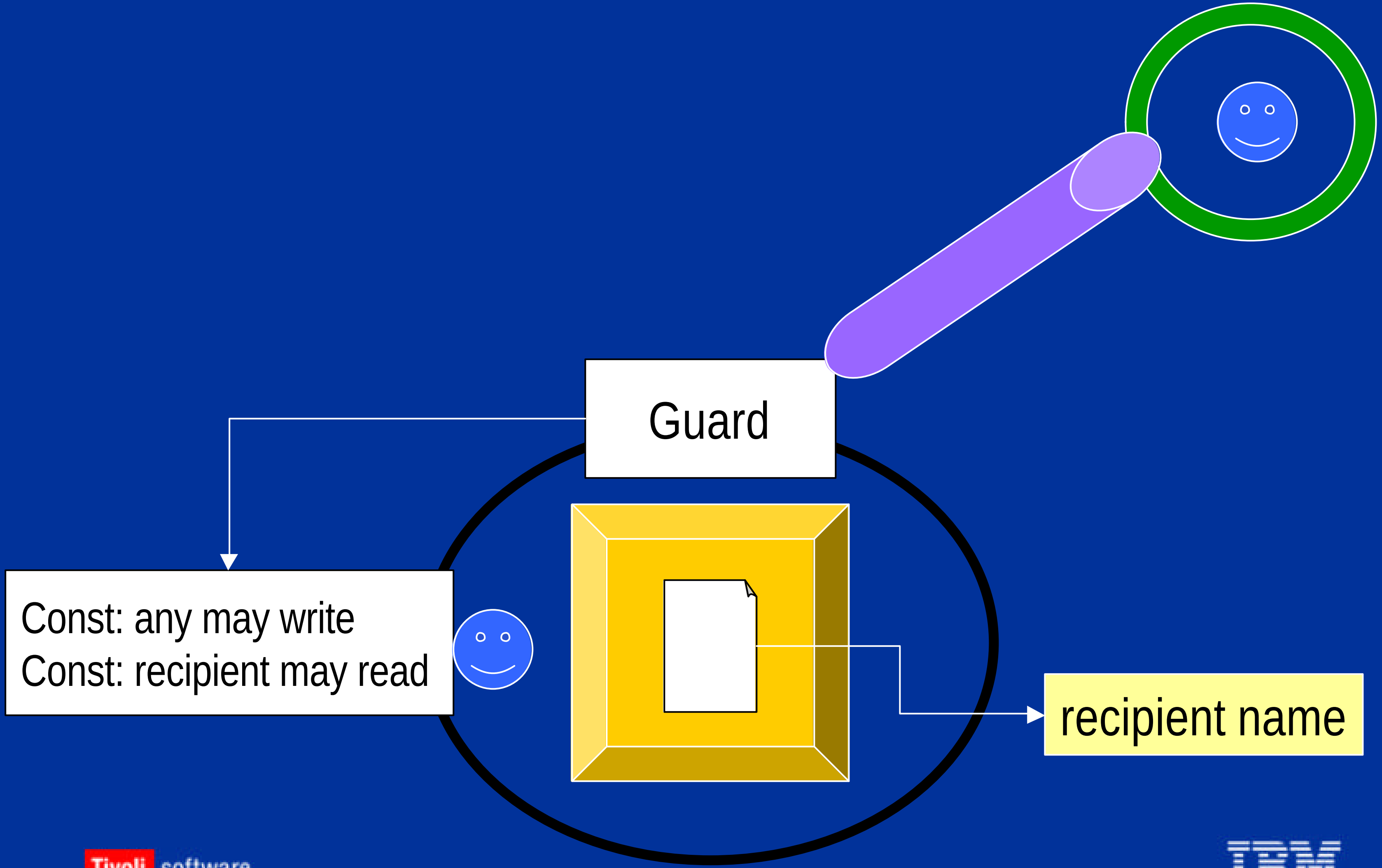
Refinement 3: Define Policy



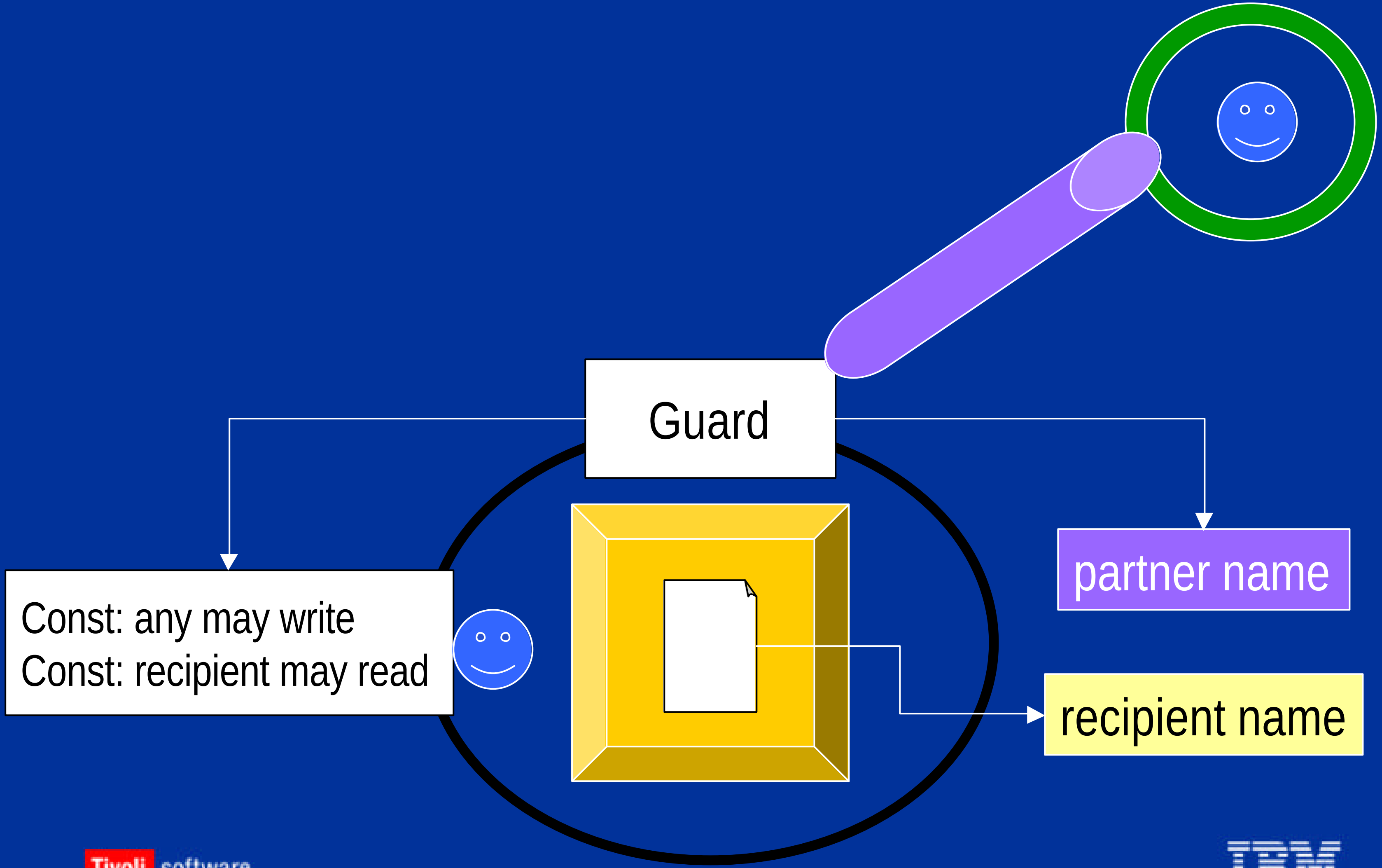
Refinement 4: Define Secure Communications


Must protect confidentiality
(constraint restricts read)
Guard
Const: any may write
Const: recipient may read

Refinement 5: Derive Target Descriptors

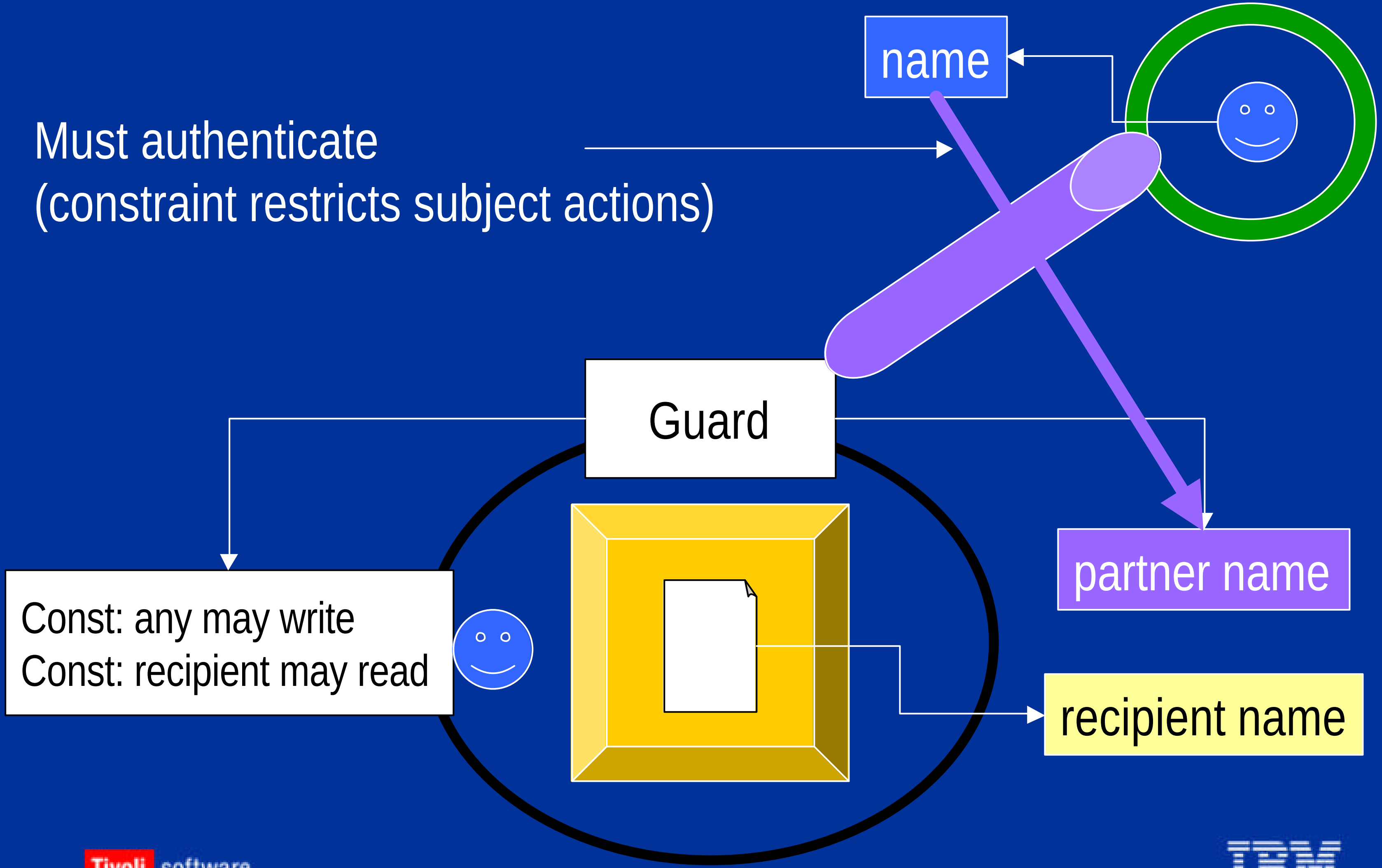


Refinement 6: Derive Security Contexts



Refinement 7: Derive Secure Associations, Subject Descriptors

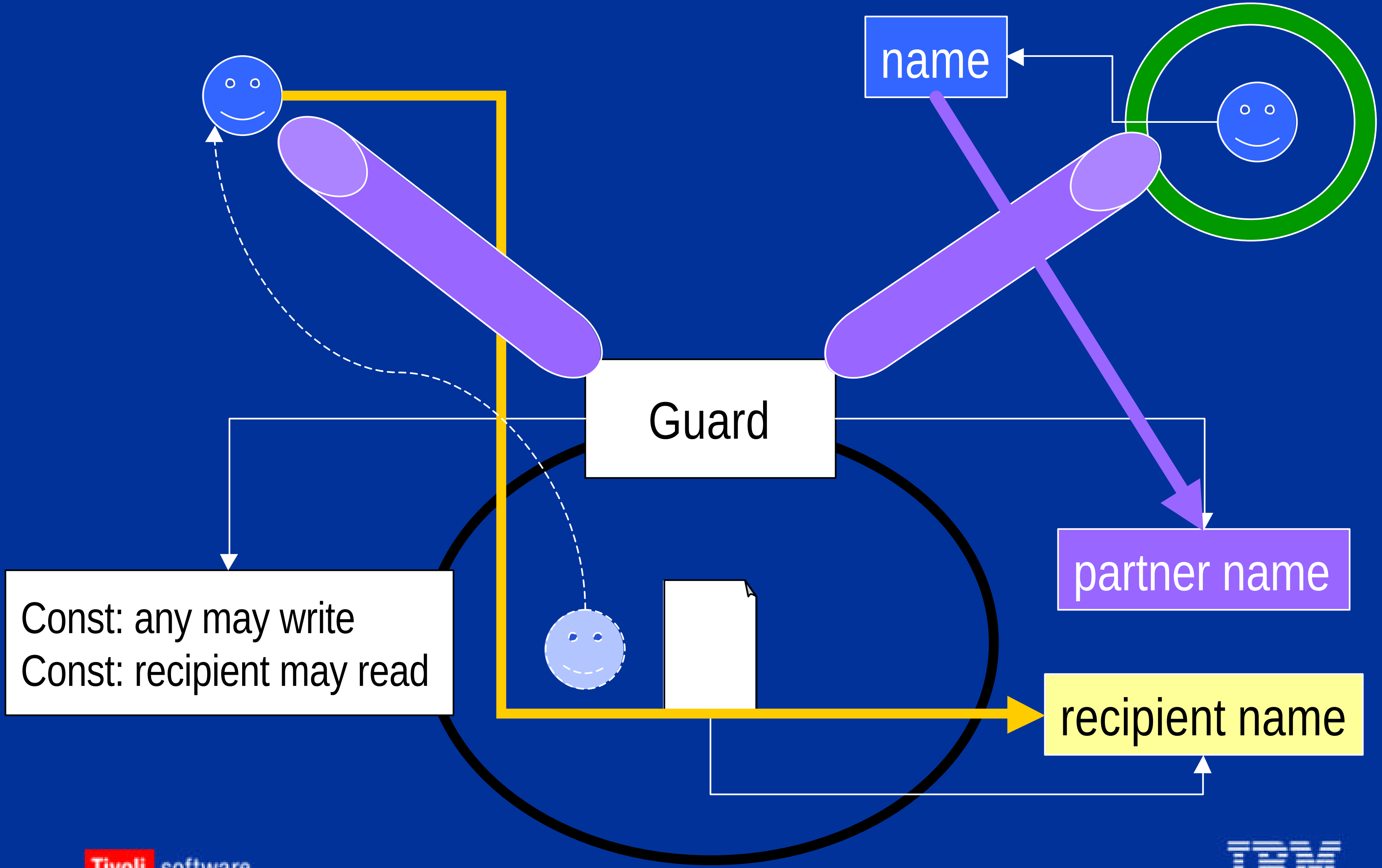
Must authenticate
(constraint restricts subject actions)



Interlude

- The system depicted is Hushmail
  - The PS boundary is enforced by encryption
  - Thus the message is inside the PS boundary even when it isn't on the sender's machine
- Problem:
  - Both parties must share a key
    - therefore the system scales as N^2
- Solution:
  - Introduce a mutually trusted party with whom all parties share a key
    - resulting system would scale as N
  - To do this, we factor the PS

Refinement 8: Consider Factoring PSs



Tivoli software

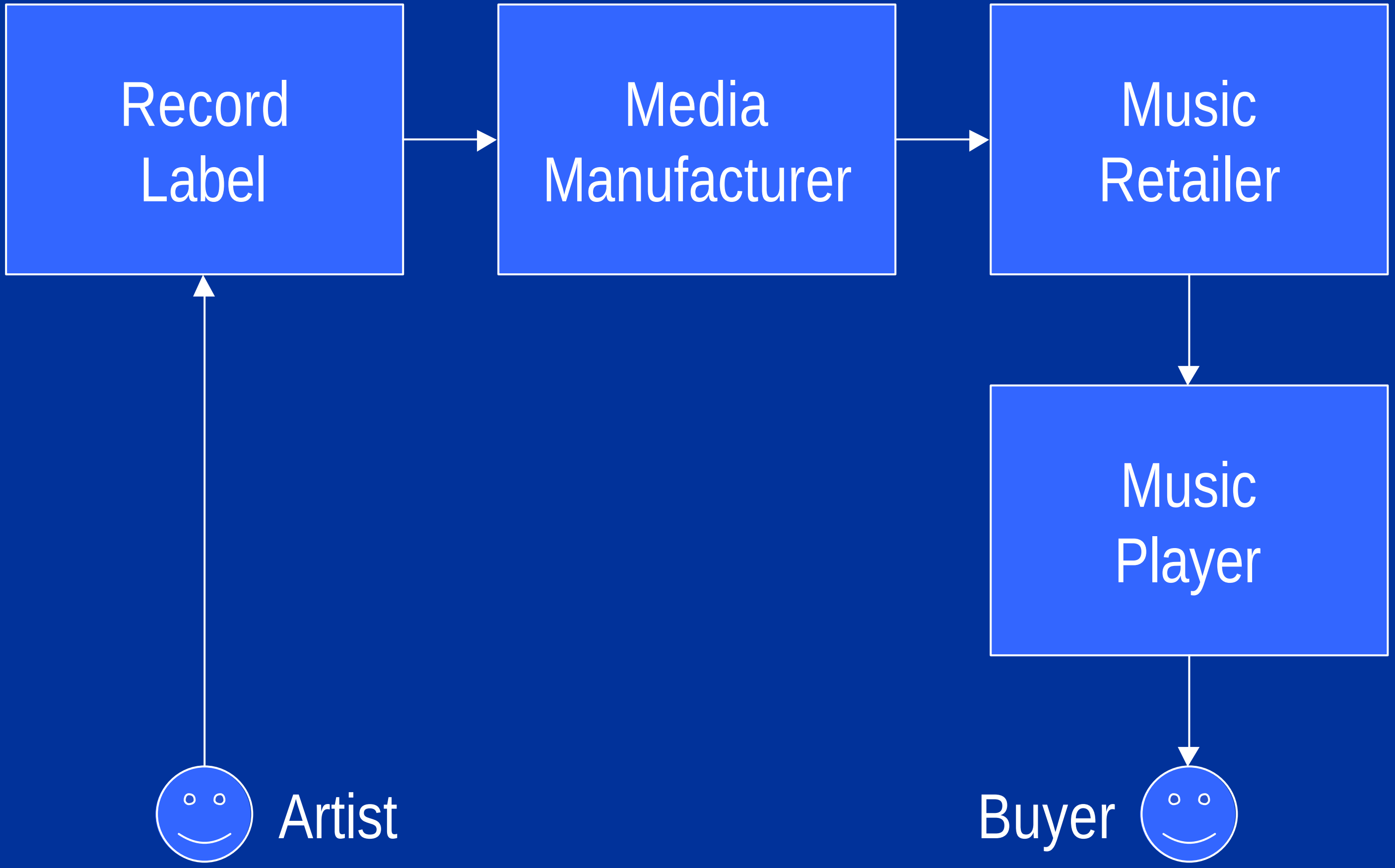
IBM

e business software

Homework Problem: Digital Music Distribution

IBM Software Group

Homework Problem: Environment



Homework Problem: Goals

- The Record Label should pay the artist for use of his music
- The Music Retailer should pay the Record Label for every sale to the buyer
- The Buyer should be able to listen to music only if he has paid