galois

Trust Relationships

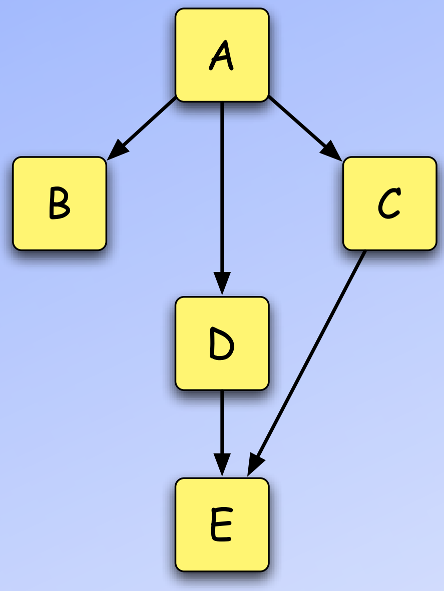
David Burke
Perry Alexander

Overview

- Defining Trust Relationships
- How they came about
- How they work
- Example & Conclusions

TR Defined

"A trust relationship diagram is a visual representation of the decomposition of security obligations throughout a system"



- What is, say, *A* trusting *B* for?
- What are the implications if that trust is violated?
- What does this diagram/analysis gain?

galois

How trust relationships came about...

Common Criteria as a Catalyst

- A couple of years ago, we began investigating EAL6 requirements
- In parallel, we were developing a "Galois High Assurance Methodology" for evaluation & certification
- Some issues with Common Criteria:
  - Boilerplate threats
  - No prioritization of threats (and associated claims)
  - No rigorous basis for strength of evidence
- Claims & Evidence at the heart of a high assurance methodology:

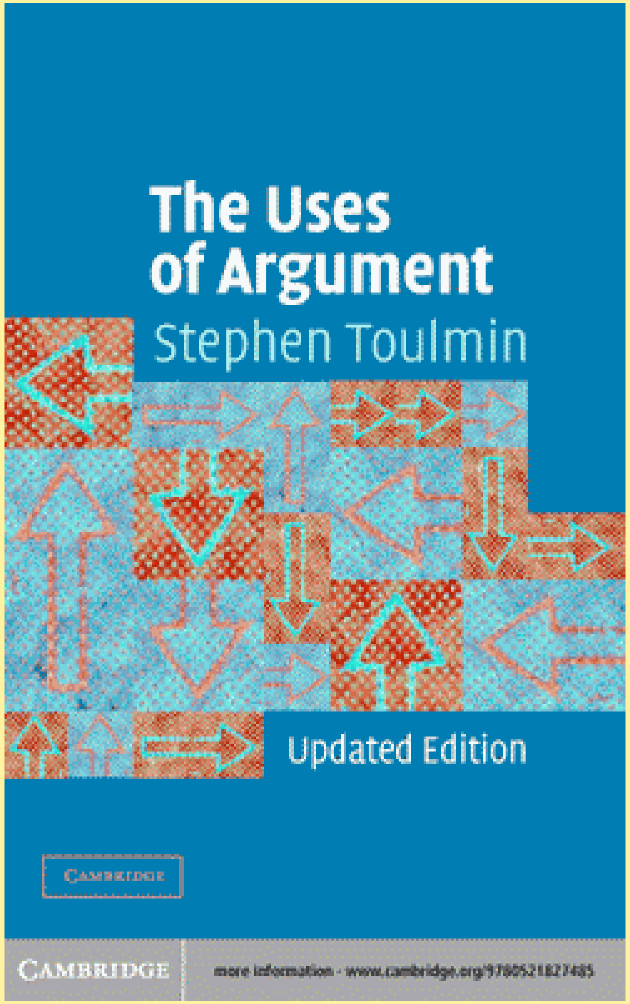
Stephen Toulmin - The Uses of Argument (1958)

Wrote "The Uses of Argument" in reaction to analytic philosophy in the mid-20th century.

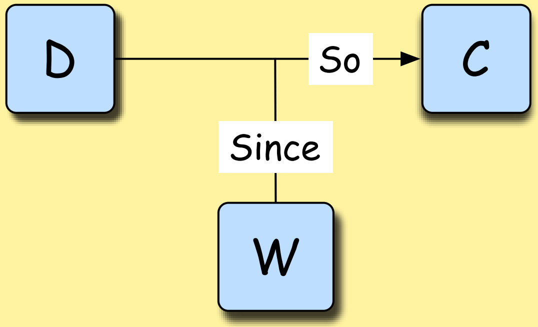
Claimed that arguments are rarely syllogisms; instead of "All A's are B's" and "No A's are B's", arguments are more likely of the form "All A's are usually B's" and "A's are rarely B's".

Invented what came to be known as the "Toulmin Model" of an argument, the heart of which is a visualization ("Toulmin Diagram") of the parts of an argument (also called a "Toulmin Structure")

This model has been adopted by scholars in fields such as law, communication, and safety engineering.



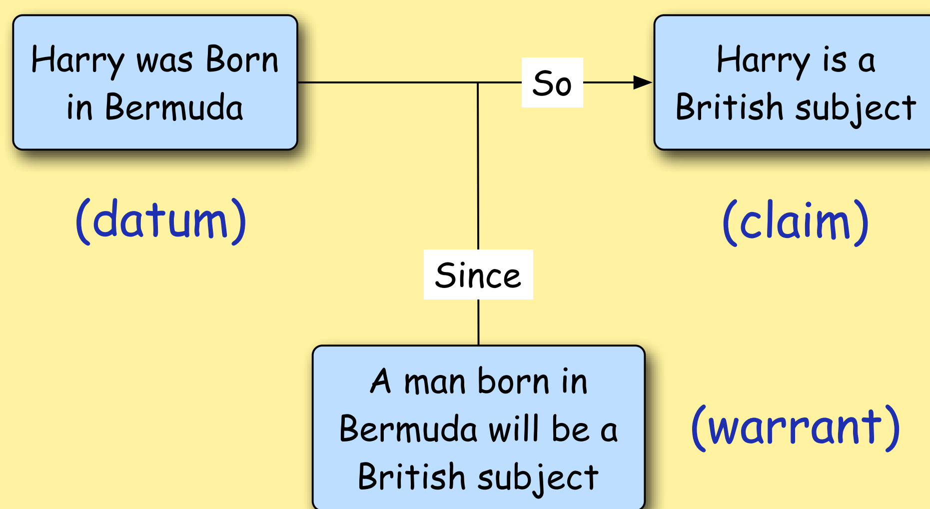
Toulmin Structures - The basic idea



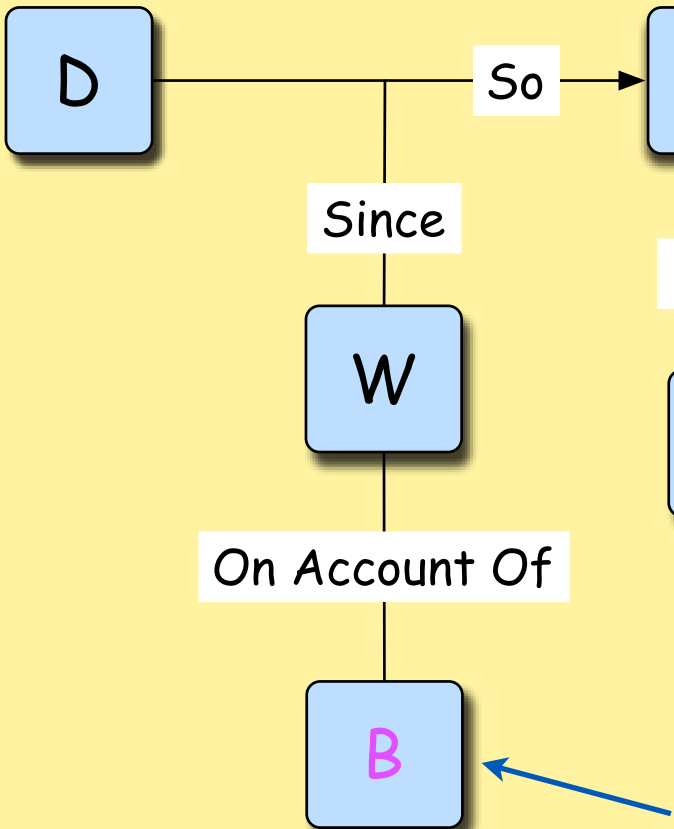
Terminology

D = datum (or evidence, observation, etc.)
C = claim (or conclusion)
W = warrant (or justification)

An example:



Toulmin Structures - Expanded



Additional Terminology

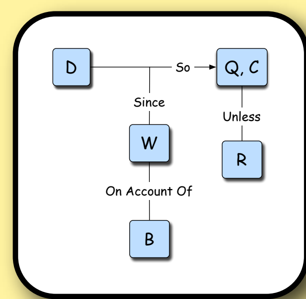
Q = qualifier

(e.g. usually, hardly ever, ...)

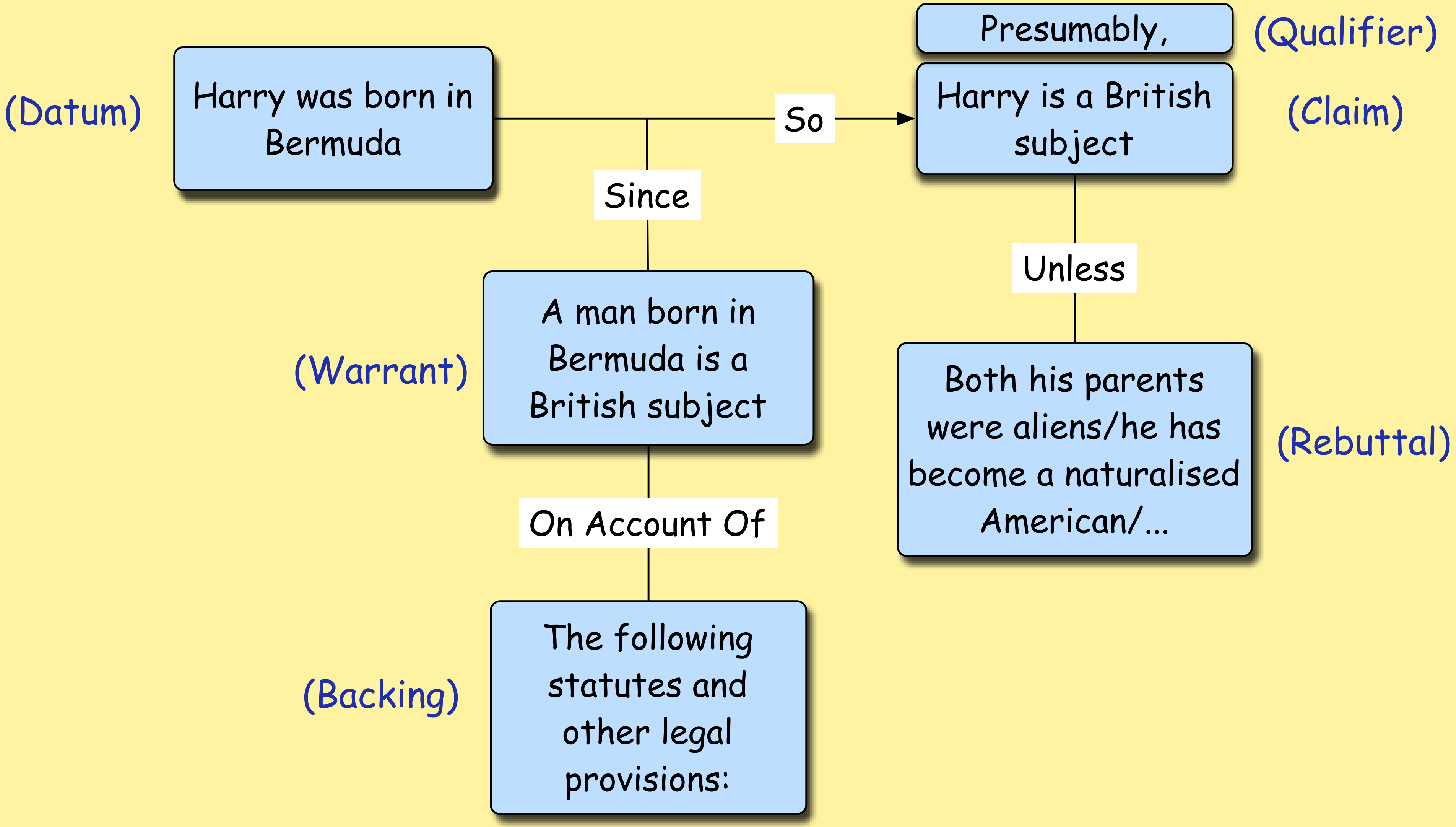
R = rebuttal

B = backing

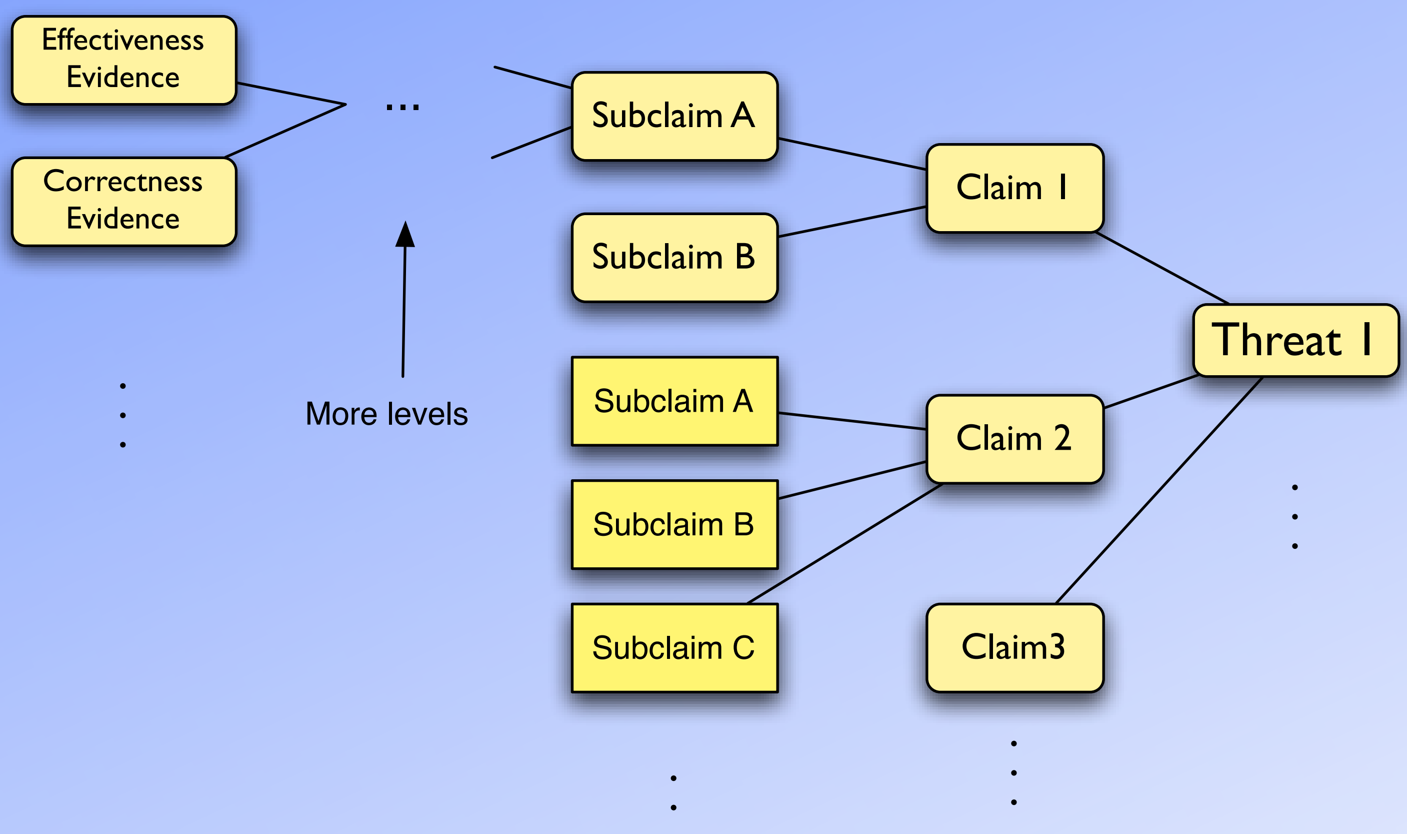
can be of the form:



Complete Toulmin Structure - Example:



Assurance Cases for Security:

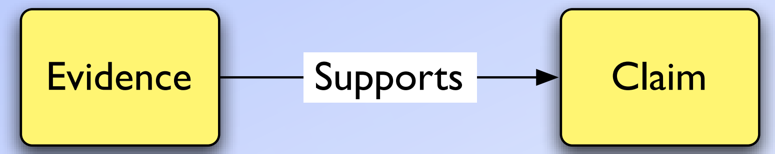


Existing criteria for evaluation of strength of evidence:

1. Relevance, Credibility, Probative Force
2. Competence, Veracity, Objectivity, Observational Sensitivity

Possible approach: convert to numerical values, and propagate up the chain...

Assurance cases provide a solid foundation for evaluating the strength of evidence, but where do the claims come from?



galois

More Catalysts - Trusted vs. Trustworthy, and "Axioms of Insecurity"

Trusted: A component is trusted <u>with respect to a security policy</u> when a system has no choice but to depend on this component to enforce the system's security policy.

Trustworthy: A component is trustworthy <u>with respect to a security policy</u> when there exists compelling evidence to a certifier/evaluator that this component adequately enforces the security policy.

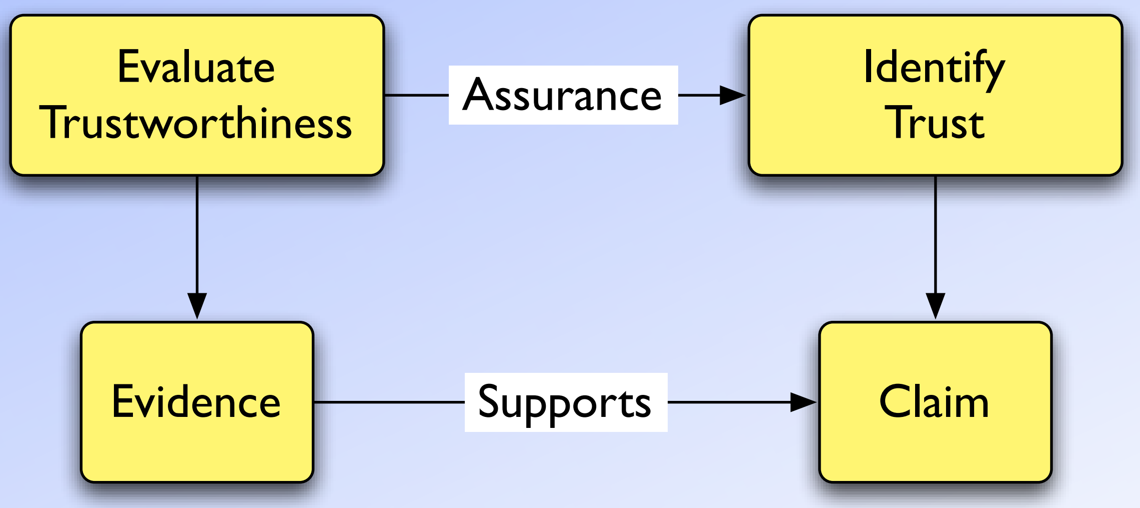
Axioms of Insecurity:

1. Insecurity exists
2. Insecurity cannot be destroyed
3. Insecurity can be moved around

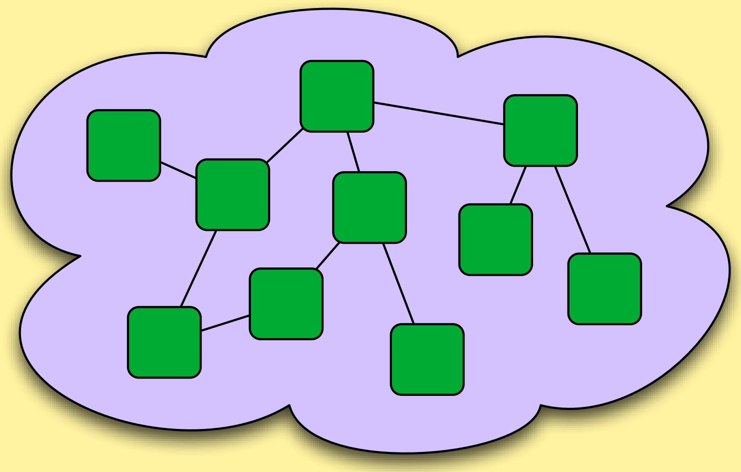
(Taken from "Trust in Cyberspace" National Academy Press, 1999)

Modeling Trust Relationships

- Make explicit where in the system we're making claims that address the threats (both type and strength)
- Visual representation - accessibility to all stakeholders (formalization is a bonus)
- Coupled with strength of evidence assessment, make risks/tradeoffs known; we don't have infinite resources to apply to assurance.



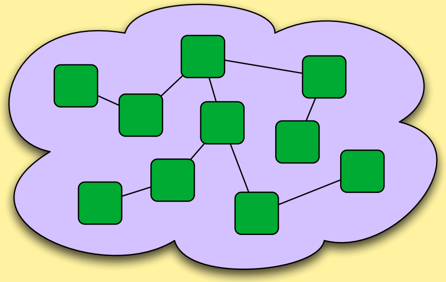
In the beginning, we have a system...



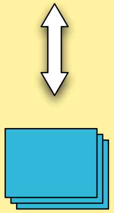
A system consists of:

1. agents or components - HW/SW/human (boxes)
2. interactions between those agents (lines)
3. a boundary - we don't consider any interactions across the boundary (cloud)

The system has a corresponding <u>security policy</u>, either explicitly, or implicitly:



System



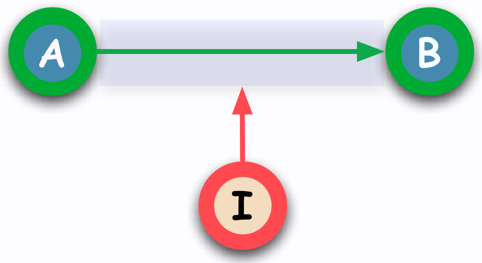
Security Policy

Security Policy:

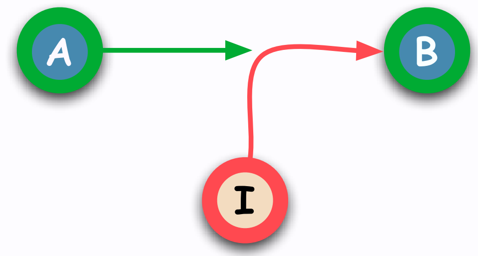
- an aggregation of all the security requirements
- in practice, a combination of top-down (customer needs) and bottom-up (existing similar architectures) analysis

A security policy addresses the following security services:

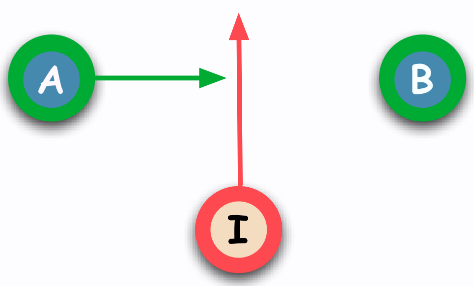
Confidentiality - protection against unauthorized disclosure.



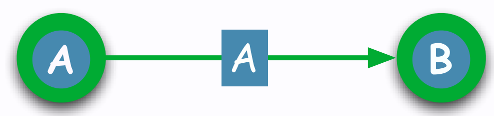
Integrity - protection against unauthorized modification or fabrication.



Availability - protection against unauthorized disruption.



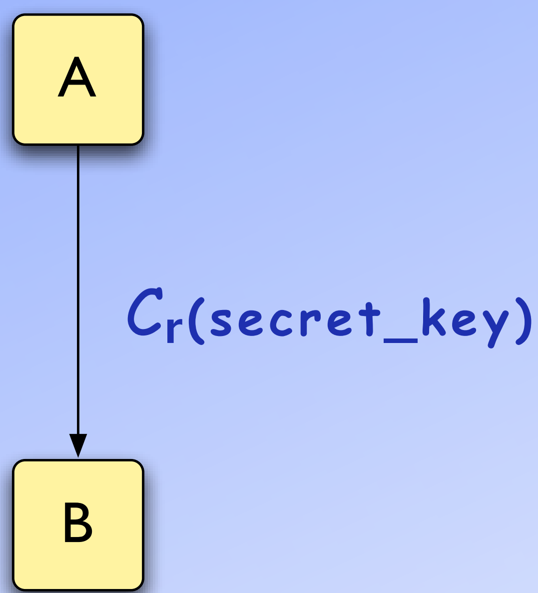
Accountability - enforcement of the bindings between agents and actions (e.g., nonrepudiation, privacy, authentication)



(arrows here represent information flows; the intruder is labeled "I")

The basic idea

C_r	Confidentiality at Rest (or storage)
C	Confidentiality of Computation (or transmission)
I_r	Integrity at Rest (or storage)
I	Integrity of Computation (or transmission)
A	Availability
N	Accountability



Agent A trusts that Agent B will keep secret_key securely stored

We could also annotate the trust arrows with a short description of what fails if the trust assumption doesn't hold.

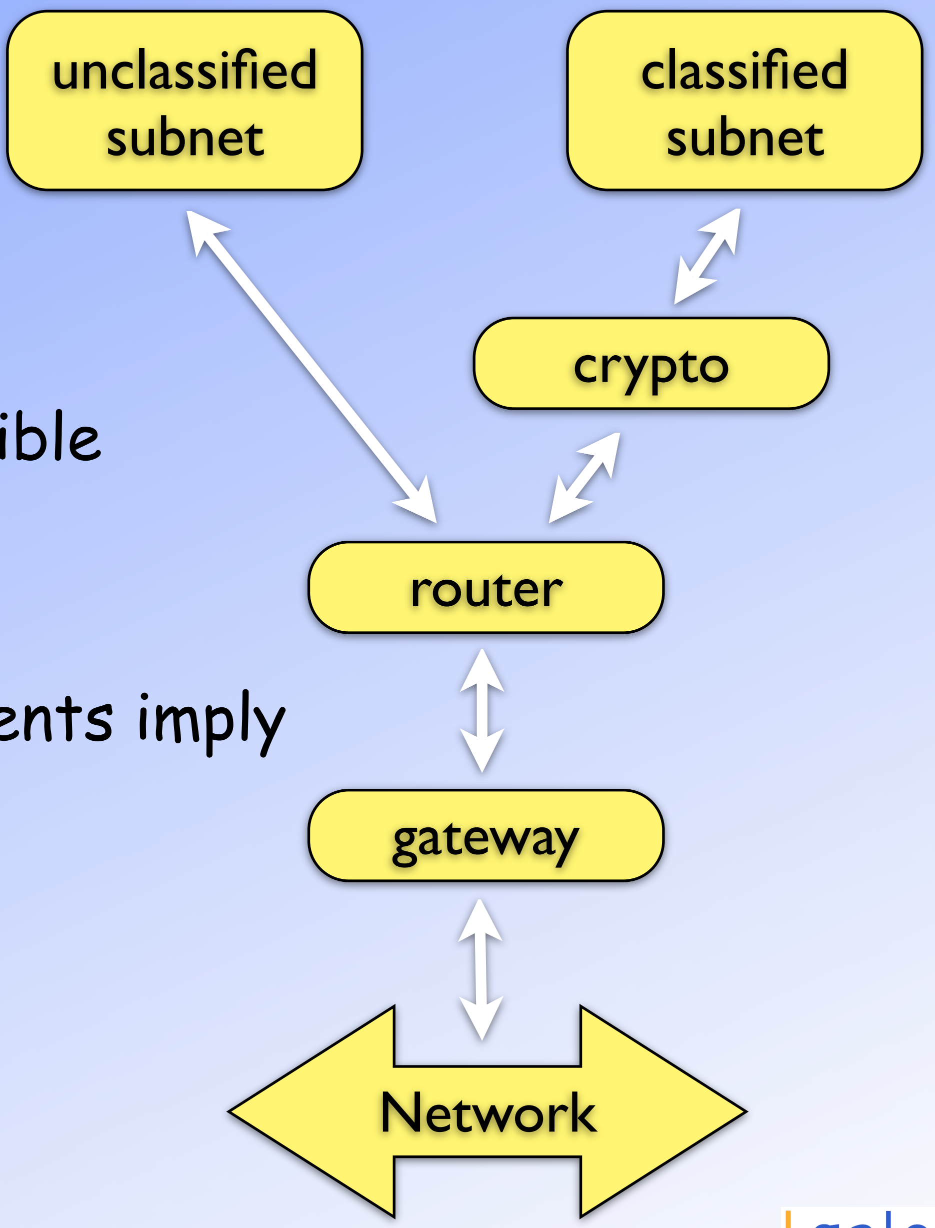
An example

Trust and System-Level Requirements

No red data on untrusted network

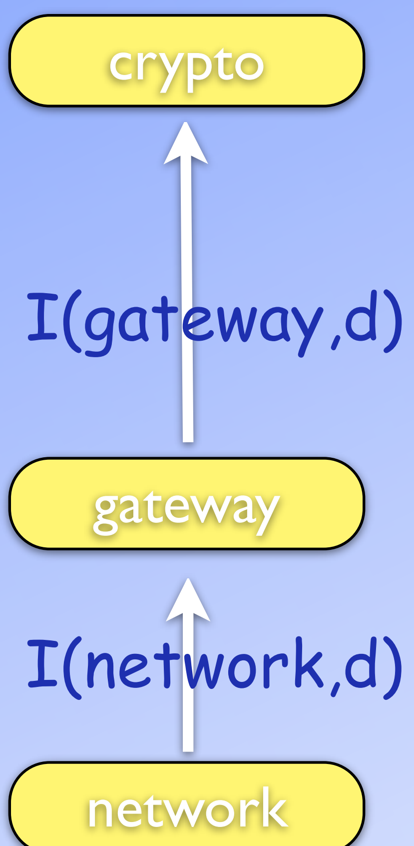
What component is responsible for enforcement?

System-level requirements imply system enforcement



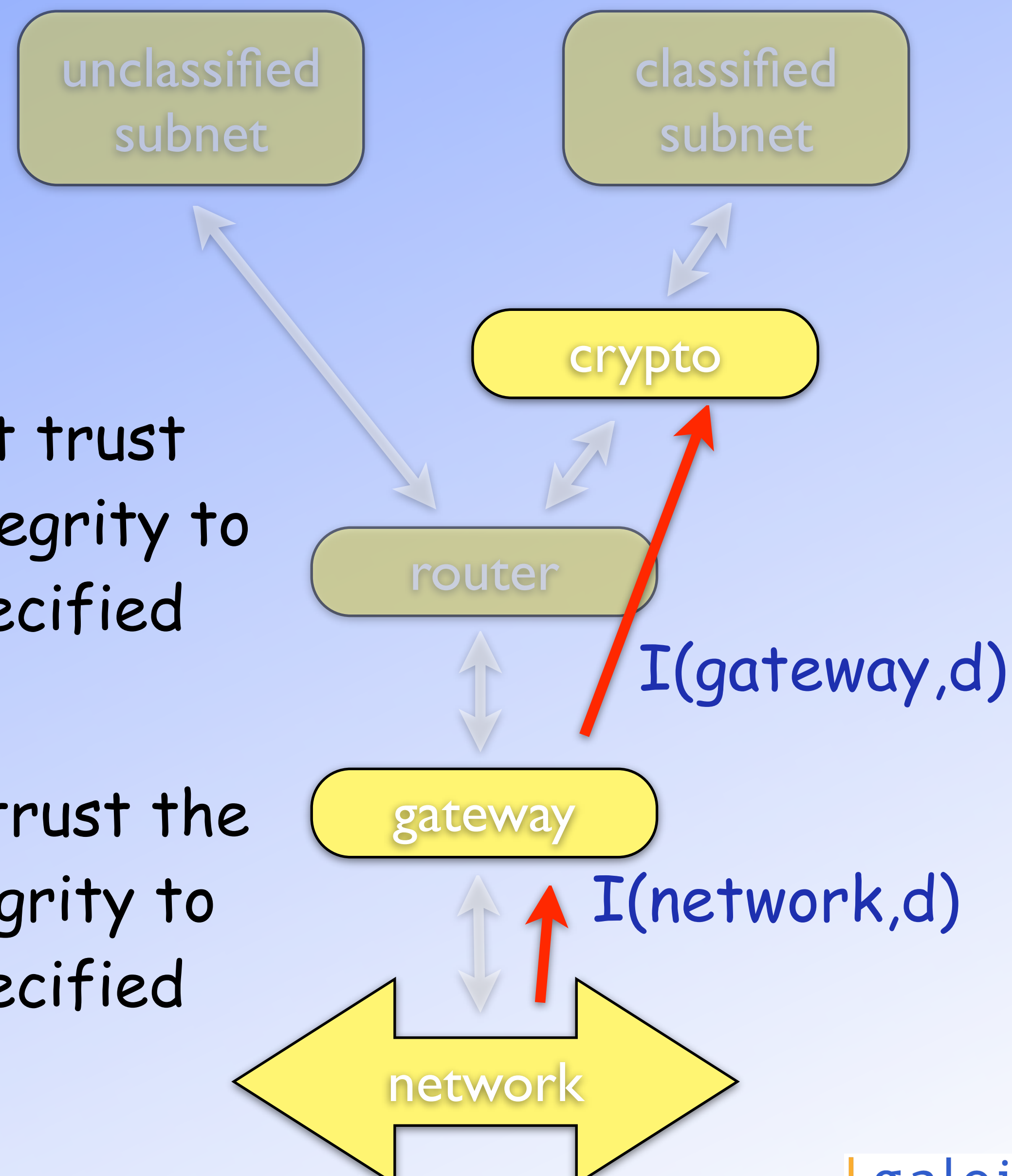
Defining a Trust Relationship Diagram

The System is responsible for the requirement



Gateway must trust encryption's integrity to behave as specified

Network must trust the gateway's integrity to behave as specified



galois

Defining Component-Level Trust

```
component crypto(secure)::static is
  black,integrity::boolean;
begin
  assumptions
    secure.integrity;
  end assumptions;
  implications
    integrity==true;
    black==true;
  end implications;
end component system;
```

Component specifications define trust assumptions and trust implications

```
component gateway(crypto::static)::static is
  black,integrity::boolean;
begin
  assumptions
    crypto.integrity;
  end assumptions;
  implications
    black==true;
    integrity==true;
  end implications;
end component system;
```

```
component network(gateway::static)::static is
  black::boolean;
begin
  assumptions
    gateway.black;
    gateway.integrity;
  end assumptions;
  implications
    black==true;
  end implications;
end component system;
```

Defining System-Level Trust

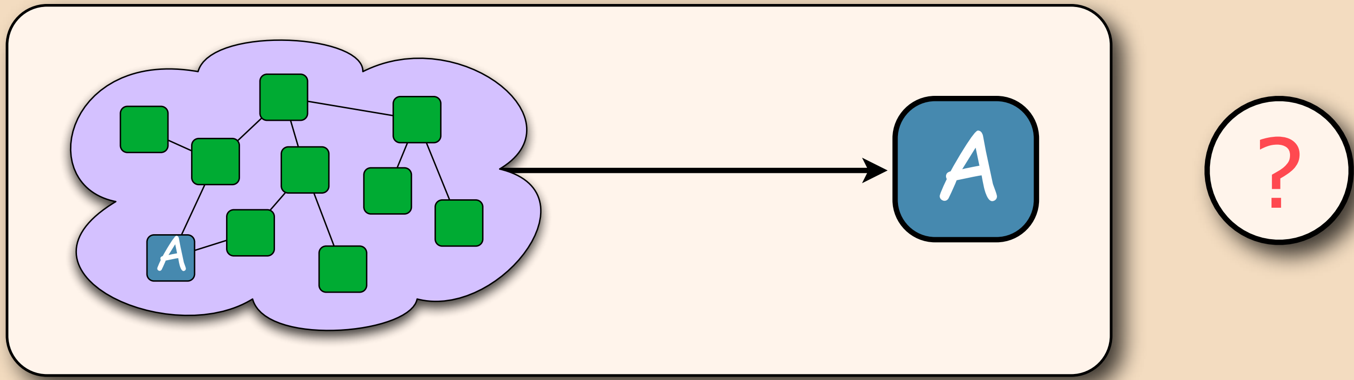
Configuration defines trust relationships between components

The system implication is that the network will carry black data

```
component system()::static is
begin
  assumptions
  end assumptions;
  definitions
    net: network(gw);
    gw: gateway(crypto);
    router: router();
    crypto: encryptor(secure);
    secure: secSubnet();
    unsecure: unsecSubnet()
  end definitions;
  implications
    net.black;
  end implications;
end component system;
```

How do I know where to draw the arrows?

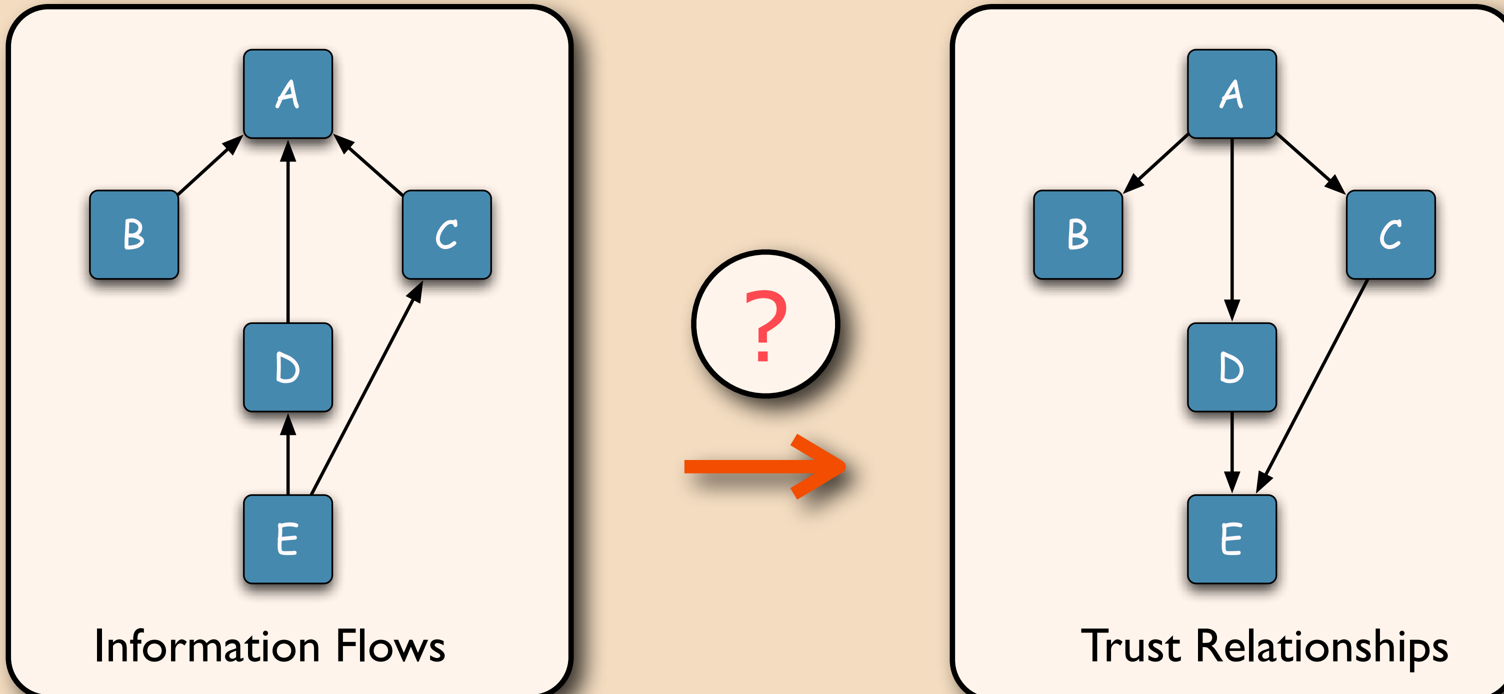
1. Isn't it the system as a whole (or the certifier, or IAO, or ISSO) that is doing the trusting of each component?



Find the component/agent which is responsible for enforcing the security policy that would be violated if the trust in A was not warranted. This is where the trust arrow should originate.

This points out the importance of Threat Modeling

2. Are trust relationship arrows constructed by simply reversing information flow arrows?



No, although it is a good place to start!

- more than one security service may be involved
- not all information flows are security-relevant

Importance of Threat Modeling!!

Backup Slides

Each Security Service has corresponding trust primitives:

$C_r(B,i)$	Confidentiality at Rest (or storage)
$C(B,i)$	Confidentiality of Computation (or transmission)
$I_r(B,i)$	Integrity at Rest (or storage)
$I(B,i)$	Integrity of Computation (or transmission)
$A(B,i)$	Availability
$N(B,i)$	Accountability

trust primitive

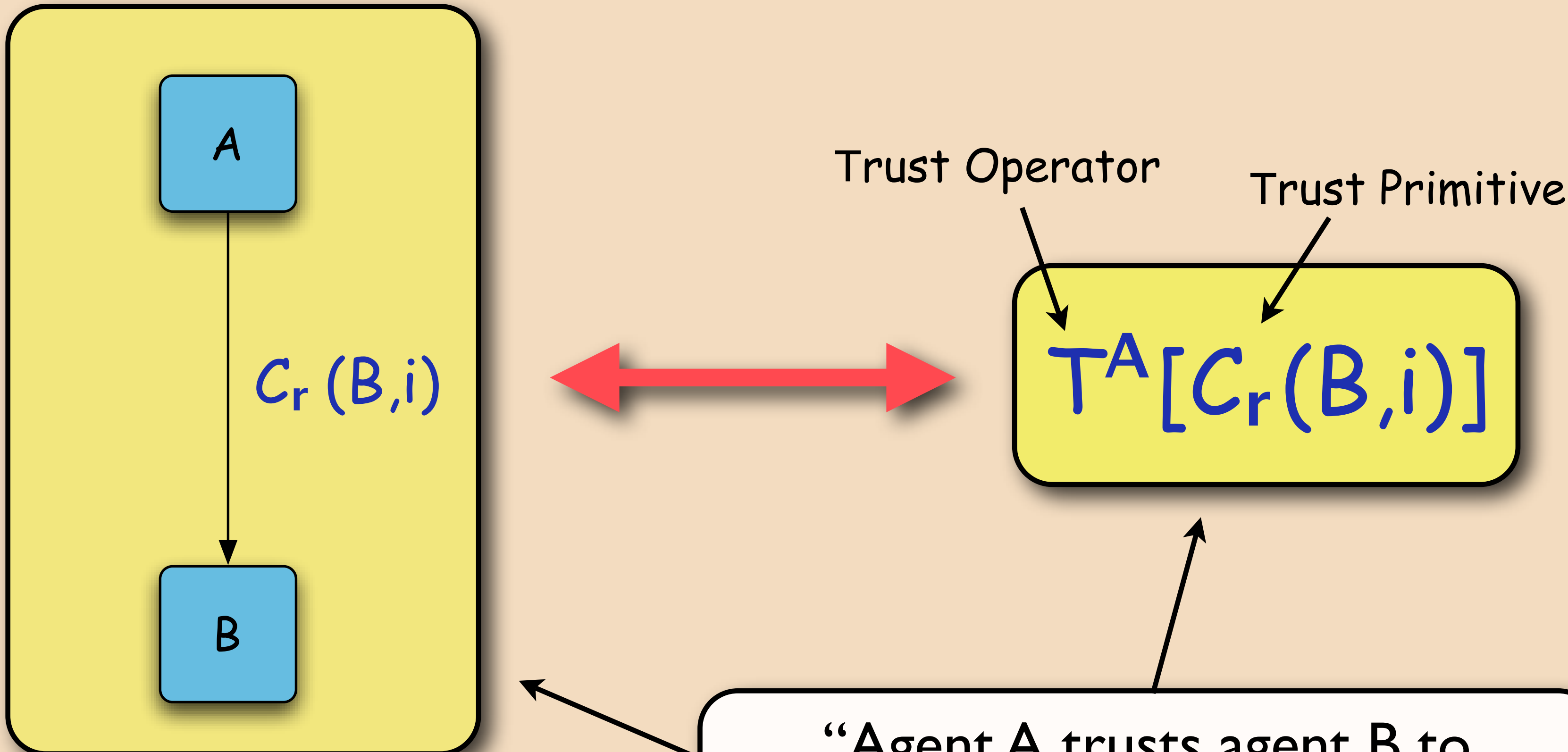
agent

data

$$C_r(B,i)$$

"confidentiality-at-rest of data i is enforced by agent B"

Semantics of Trust Relationships - Example



"Agent A trusts agent B to enforce the confidentiality-at-rest of data i"