Scenarios for Security based on Proof Carrying Code

Germán Puebla

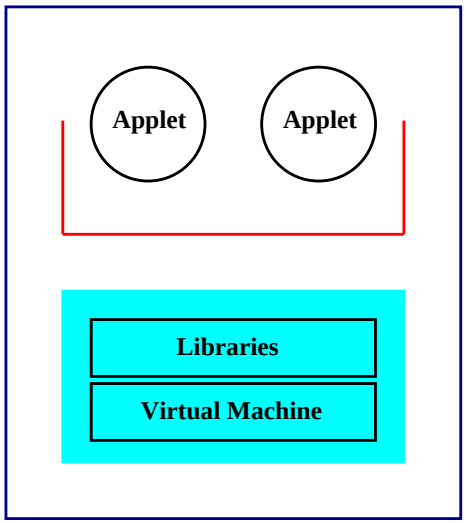
Universidad Politécnica de Madrid (UPM), Spain

April 28th, 2006

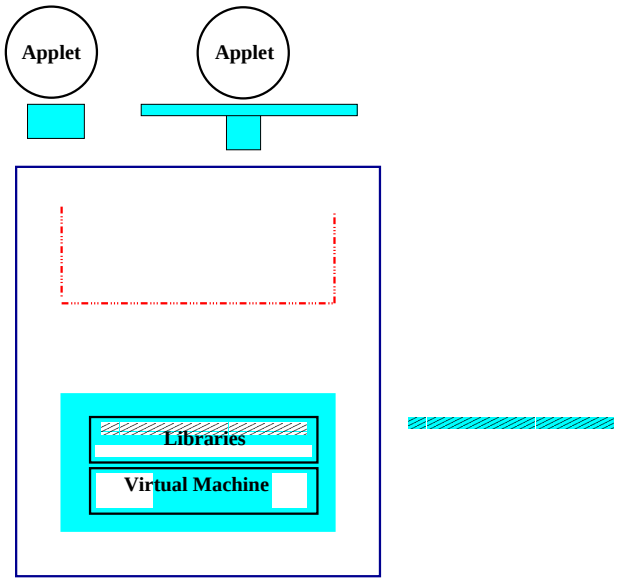
Computational model

Very large networks of JVM-enabled devices:

- No central trust authority: trust infrastructures must allow verifiable evidence (cryptography is not enough)
- Devices must host extensible computational infrastructures that can be updated remotely
- Sanboxing is not enough. No sharp distinction between static Trusted Computing Base and mobile applications



Ideal scenario



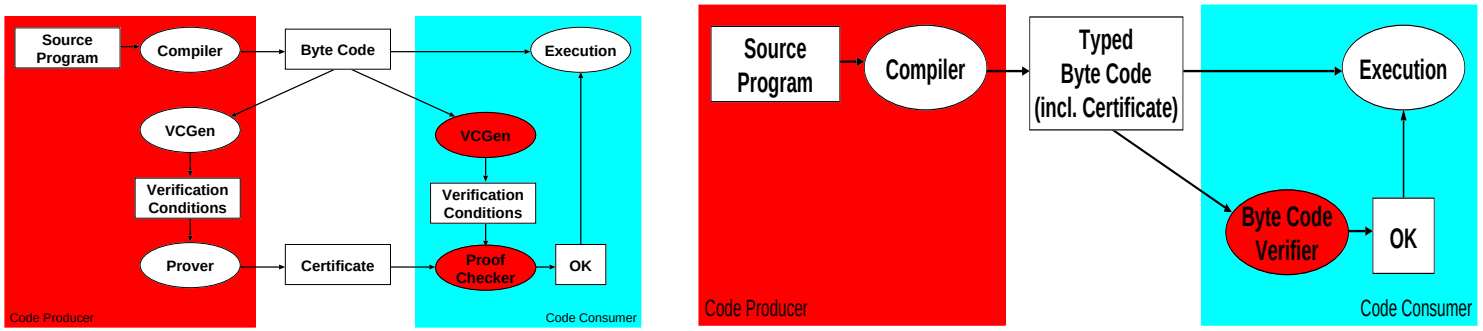
Scenario considered

Security issues

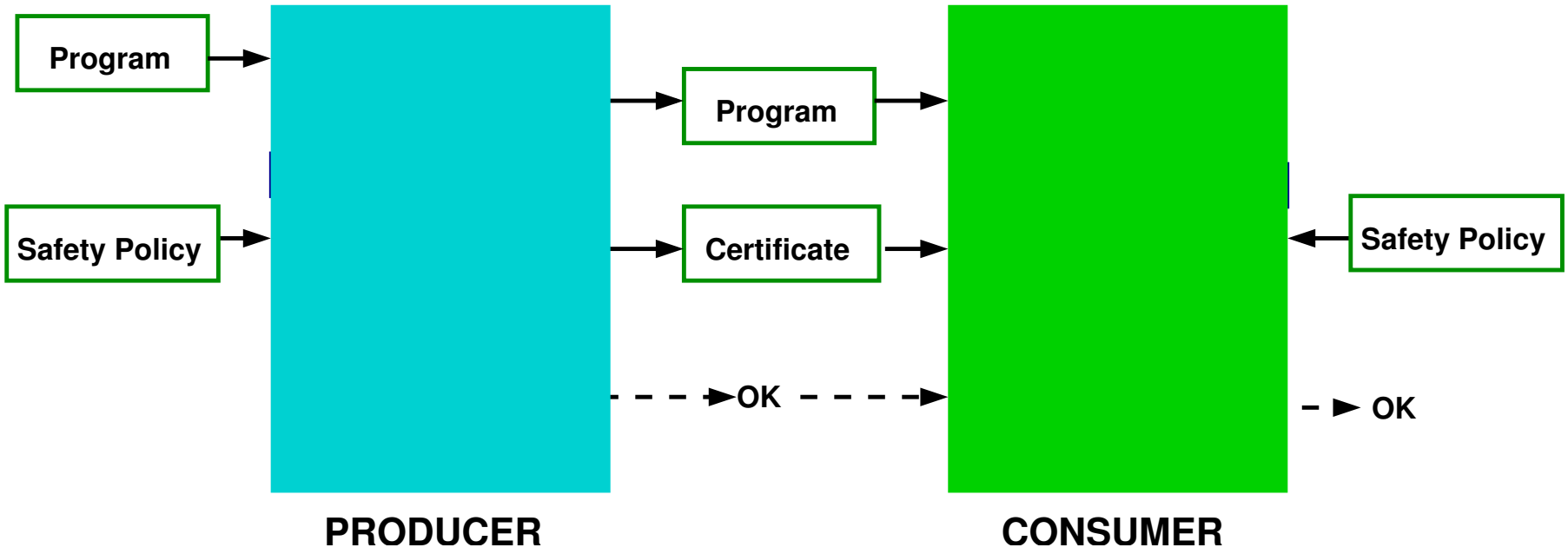
- Devices must be protected:
  - in a static and proactive way
  - individually
  - using resource aware enforcement mechanisms
- Need for expressive security policies and functional specifications:
  - Information flow and resource control policies
  - Framework-level policies and application-level policies

Security framework

To deliver a framework with appropriate characteristics, MOBIUS adopts ideas from *Proof Carrying Code* (PCC) and require that downloaded components come equipped with certificates.



Certificate-based Mobile Code Safety



Proof-Carrying Code (PCC)

- *Proof Carrying Code* (PCC) is a general technique for mobile code safety which associates safety information (*certificates*) to programs.
  - *Producer:* Generates a certificate (or proof) at compile time by using a *certifier*. Then, submits it to the consumer.
  - *Consumer:* Receives (or downloads) the untrusted package "program + certificate". Then, runs a *checker* to verify compliance with the safety policy.
- Key benefit: burden of ensuring compliance with desired safety policy (mostly) shifted from consumer to supplier.
- Since the checker is smaller than the certifier, the TCB is reduced.

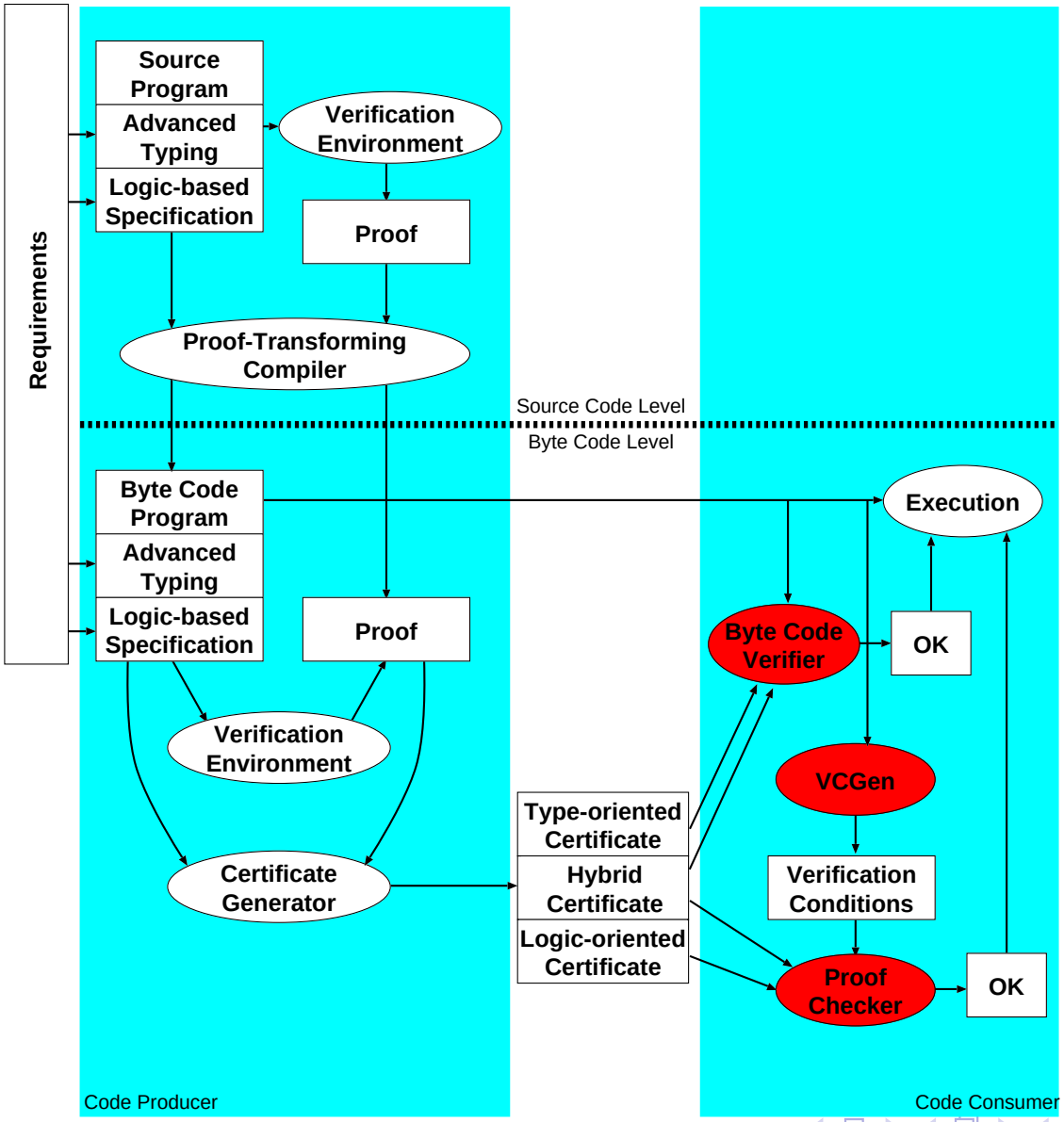
Proposals and Challenges

- Some efficiency considerations
  - Checking is performed by every consumer (vs. certificate generation once).
  - Receiver could be a small embedded system with limited resources.
- Fundamental challenges:
  1. defining *expressive safety policies* covering a wide range of properties,
  2. obtaining *easy-to-use certificate generators* and,
  3. designing *simple, reliable, and efficient checkers* for the certificates.

Issues

- The *Enabling technologies*: should address the three issues mentioned.
- Enabling technologies: should provide enough precision and automation to guarantee applicability and scalability.
- MOBIUS is set out to develop techniques that draw from and tightly combine both automatic and interactive technologies, including:
  - Type systems:
    - efficient, automatic, but specialized and imprecise
    - used for information flow, resource usage, aliasing
  - Program logics: general, precise, but interactive
    - characterization of non-functional properties
    - characterization of high-level security policies
    - component correctness

The MOBIUS Framework



Venues for Synergy

- Fundamental programming language technology:
  - Design of security policies.
  - Program verification.
  - Static analysis techniques.
    - type systems
    - abstract interpretation
  - Automated deduction
- Security infrastructure for Java/JVM:
  - Tools and enviroments for reasoning about Java and Java bytecode programs.
  - Compile-time and run-time verification in virtual machines.
  - Security architectures, APIs, etc.
  - Including adaptive security.
- Extending the PPC to other scenarios
  - Multiple producers/consumers.
  - Mobile code transformation, adaptation.
  - Combining PPC with trust.