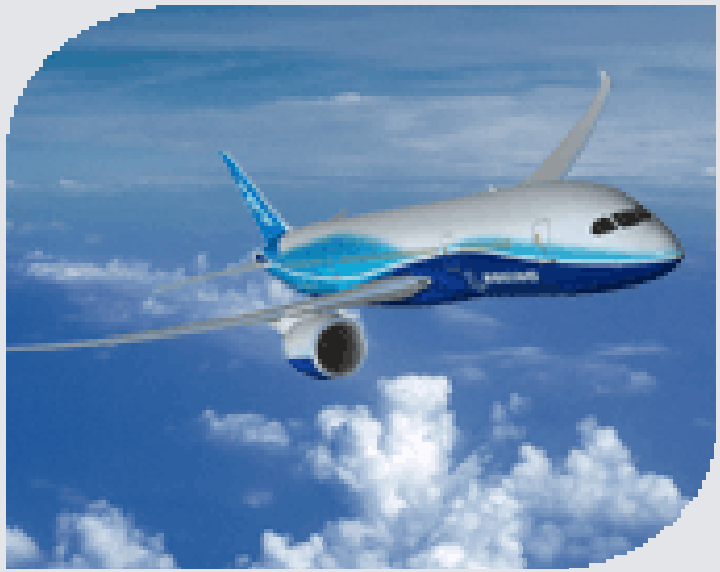
Formal Security Analysis of Electronic Software Distribution Systems

David von Oheimb [1]
Monika Maidl [1]
Richard Robinson [2]

[1] Siemens Corporate Technology, Munich
[2] Boeing Phantom Works, Seattle



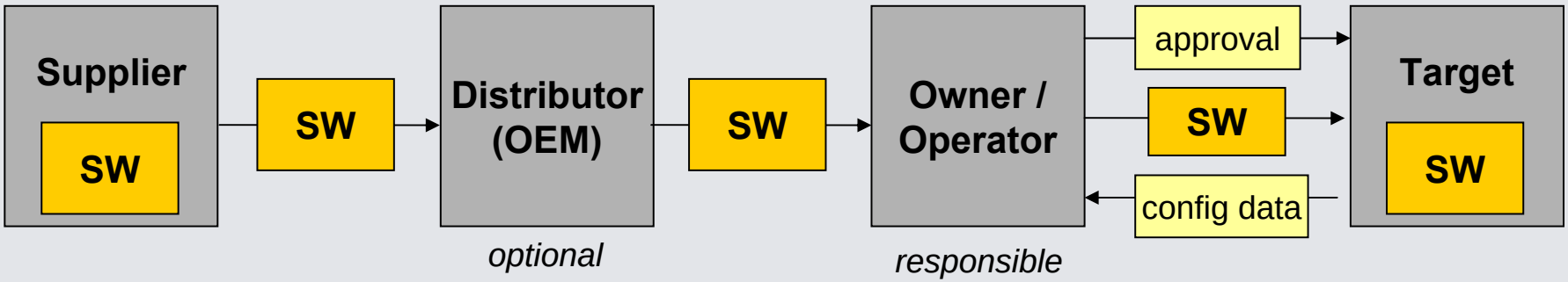
SAFECOMP 2008 Newcastle, 24 Sep 2008

Overview

- Security requirements of a Software Distribution System (SDS)
- Architecture for a Software Distribution System
- Compositional security assessment
- Conclusion

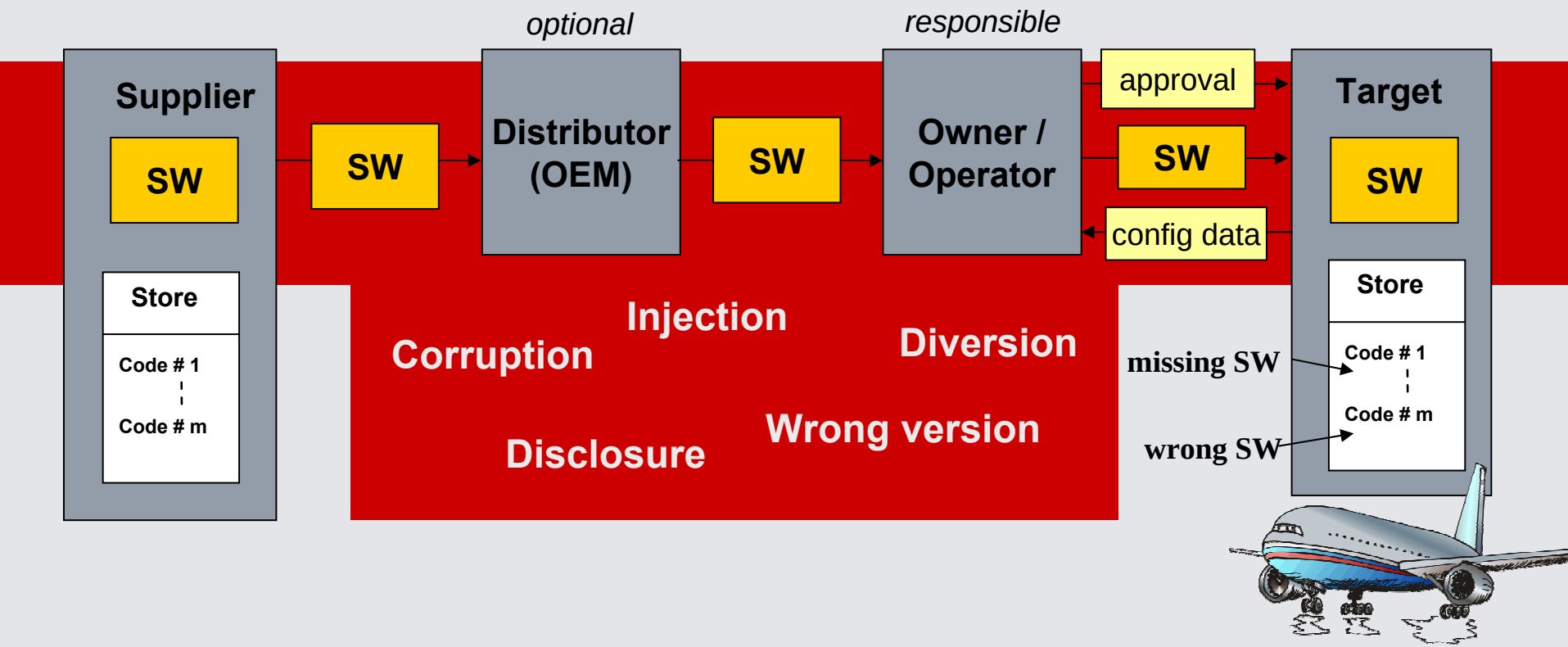
Motivation for Electronic Software Distribution Systems (SDS)

- IT system with **networked devices** in the field performing **safety-critical** tasks:
- Safety (and/or security) of system depends on **secure update** of embedded software in field devices.
- Examples: Airplanes, automobiles, energy distribution, digitally programmed machine tools, medical devices …



Transition from media-based (CD-ROMs etc.) to **networked transport** increases **security risks** due to transport over open, insecure networks.

Security Threats



Internal attacks **Attacks via open network** **Internal attacks**

Security Objectives

BOEING

SIEMENS

end-to-end

- **Authenticity**: Every software item accepted must originate from a genuine supplier.
- **Integrity**: A software item accepted at a target, its identity and contents must not have altered during transports.
- **Confidentiality**: Software items must be kept secret from entry point until reaching the target.

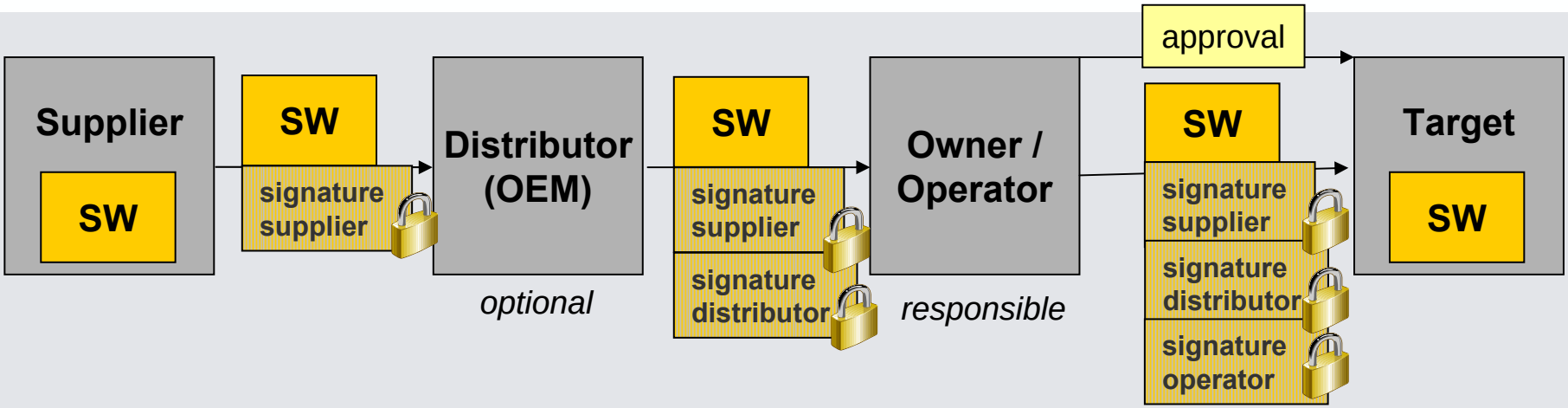
last hop

- **Correct Destination**: A target device must accept only software items for which it is the destination intended by the operator.
- **Correct Version**: A target device must accept software items only in the latest version approved by the target operator.

Overview

- Security requirements of a Software Distribution System (SDS)
- Architecture for a Software Distribution System
- Compositional security assessment
- Conclusion

Architecture for a Software Distribution System

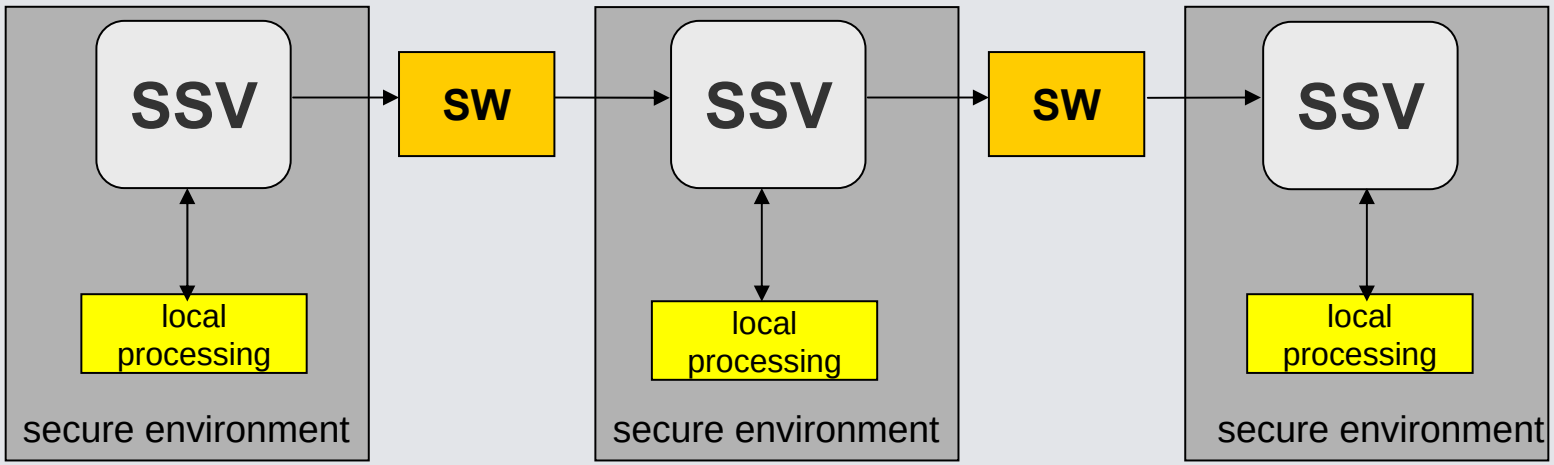


- Intermediaries might just forward SW, or do local processing (quality checks, setting target-specific parameters, inserting licensing keys).
- **Digital signatures** are the basic mechanism used to protect software:
  - sender applies private key to SW,
  - the receiver performs signature verification with corresponding public key

  => **authenticity of sender and integrity during transport**.
- Every intermediary checks the signatures and might add a new one.

Software Signer Verifier (SSV)

Signature generation/verification is the central security function each node in the SDS has to perform. Bundling in a special component, the **Software Signer Verifier**, makes a SDS more modular and flexible.



- **Introducing** software into SDS by signing it.
- **Checking authentication and authorization** of the sender by verifying the signature on software received from another SSV instance.
- **Approving** the SW by adding a signature.
- **Delivering** software out of the SDS after successfully verifying it.

Overview

- Security requirements of a Software Distribution System (SDS)
- Architecture for a Software Distribution System
- **Compositional security assessment**
- Conclusion

Common Criteria (CC) for IT security evaluation

www.commoncriteria.org

Product-oriented methodology

for **IT security assessment,**

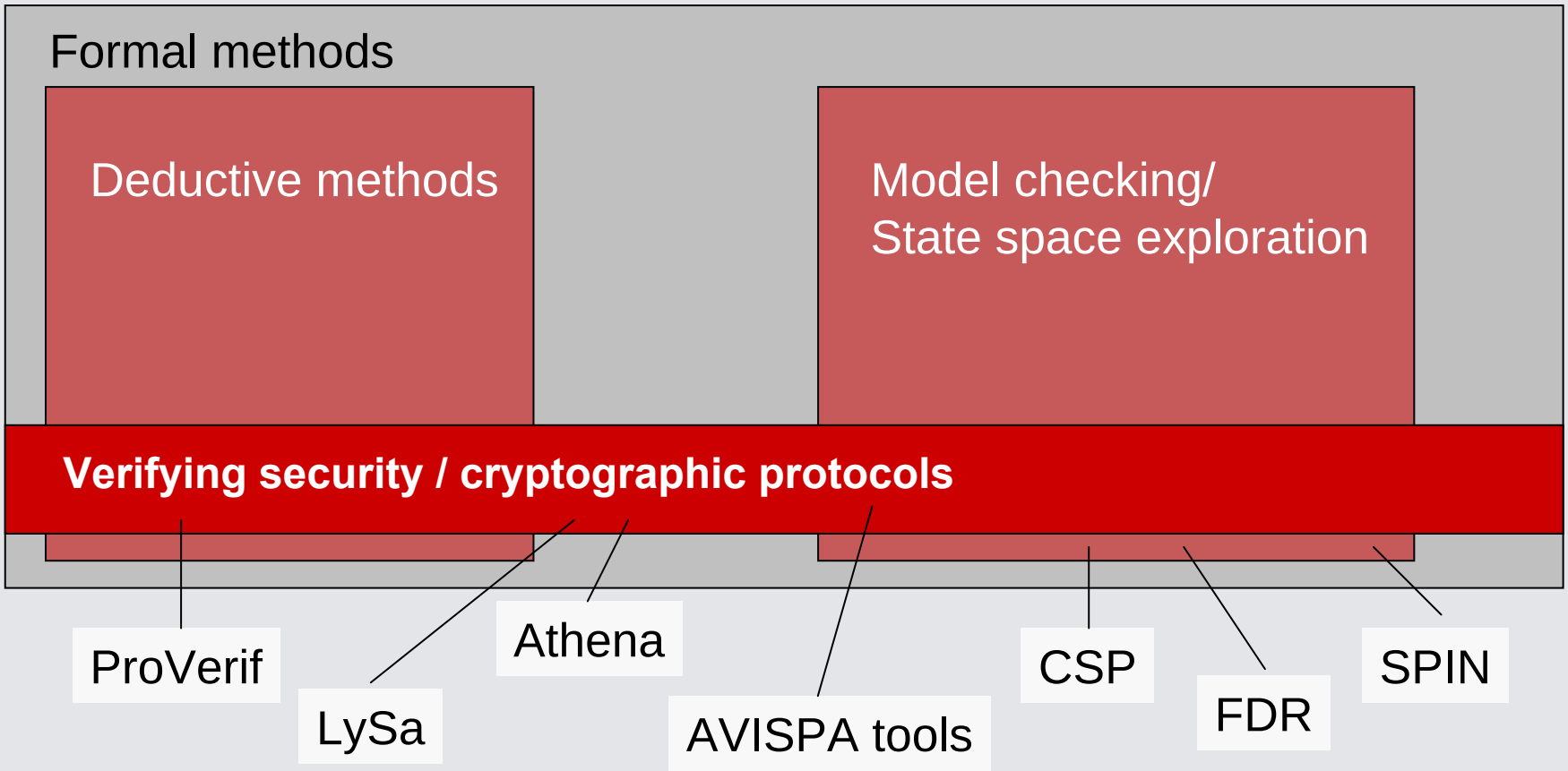
ISO/IEC standard 15408

- Approach:
  - -What are the **objectives** the system should achieve?
  - -Are the **measures** employed appropriate to achieve them?
  - -Are the measures **implemented and deployed correctly**?
- Assessment with different level of depth and rigor: EAL 1 – 7
  - -Level 6 and 7 require formal models.
  - -Highly safety-critical systems, e.g. avionics, might require EAL 6.

Applying Common Criteria to SDSs

- SDSs are complex distributed systems with many components, while CC are product oriented.
- Assessment of a complex system very costly and time-consuming.
- CC offer only limited support (“CAP”) for compositional system assessment, in particular not above EAL 4.

Formal analysis

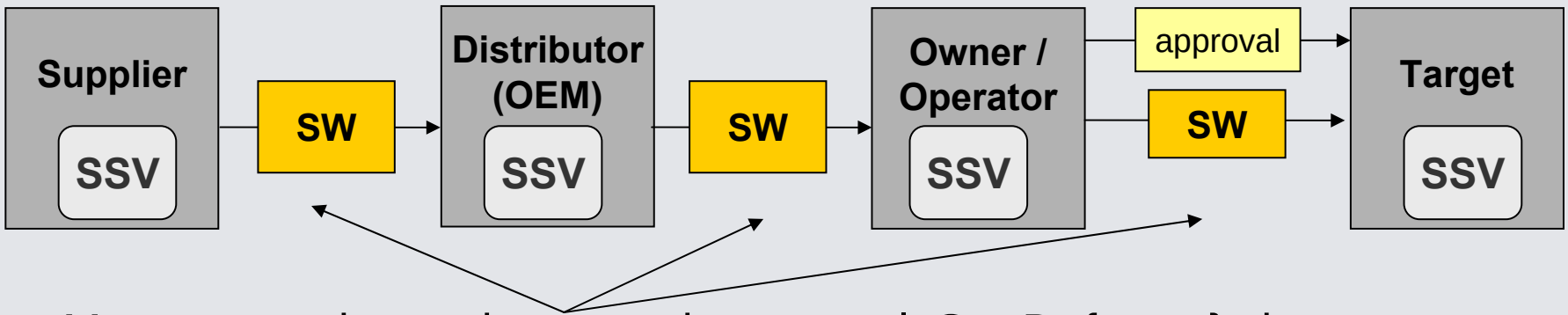


Restricted to high-level systems due to state space explosion.

Compositional security assessment

Evaluation of the **SSV according to Common Criteria** EAL4 (non-formal).
Local security objectives are defined in a CC protection profile and validated according to the CC methodology.

Analysis of the **interaction/message exchange of SSVs in a SDS** in a formal way. Security objectives of SSVs are incorporated as assumptions of the high-level security protocol.
System security objectives are proven with AVISPA model checker.



Message exchange is a security protocol: S-> D: {mess}:sig, …

Compositional security assessment

- **Pragmatic approach**, offering a good cost/benefit ratio.
- **Modularity**: Clear interface between SSV and system
  - SSV can be replaced by different implementation.
  - Architecture of SDS can be changed while keeping SSVs as components.

Formal modelling of SDS

1. SUP -- Software.{h(Software).DIS}_inv(KSUP).CertSUP --> DIS

2. DIS -- Software.{h(Software).DIS}_inv(KSUP).CertSUP
.{h(Software).OP }_inv(KDIS).CertDIS --> OP

3. OP -- Software.{h(Software).DIS}_inv(KSUP).CertSUP
.{h(Software).OP }_inv(KDIS).CertDIS
.{h(Software).TD }_inv(KOP).CertOP --> TD

SUP: software supplier with private key inv(KSUP)

DIS: software distributor with private key inv(KDIS)

OP : target operator with private key inv(KOP)

TD : target device

Signatures comprise hash value of software and **intended receiver.**

Signatures are applied in parallel. CertN is a certificate for the public key of N (signed by Certification Authority or self-signed).

Overview

- Security requirements of a Software Distribution System (SDS)
- Architecture for a Software Distribution System
- Compositional security assessment
- Conclusion

Conclusion

- Architecture and security objective for safety-critical distribution of software with intermediaries.
- Generic SSV component that is instantiated at different nodes of a SDS.

- Security assessment by combining Common Criteria with formal methods.
  - Common Criteria most widely accepted methodology, but not targeted at systems composed of instantiations of a generic component.
  - Formal methods work well for high-level security protocols, but suffer from state explosion when applied to implementations.
  - Combining the methods according to their strengths, and with good cost/benefit ratio.

- Future steps
  - **Trust management** aspects including Public Key Infrastructure (PKI).
  - **Configuration management** with installation instructions and reports.

BOEING

SIEMENS

Thank you for your attention.

Questions?