Do You Know Where Your Data Are? Secure Data Capsules for Deployable Data Protection

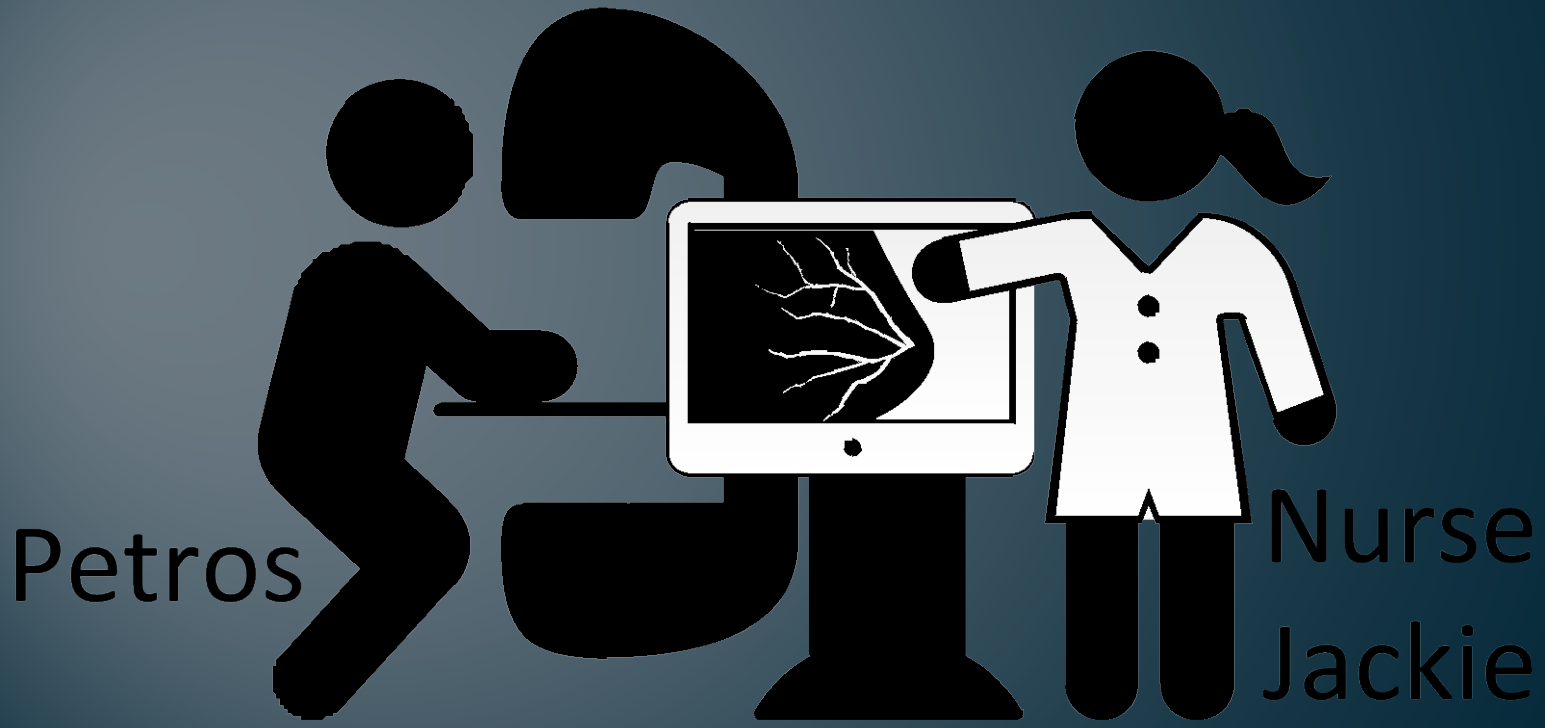
Petros Maniatis, Devdatta Akhawe,
Kevin Fall, Elaine Shi,
Stephen McCamant, Dawn Song

Stanford Clinic

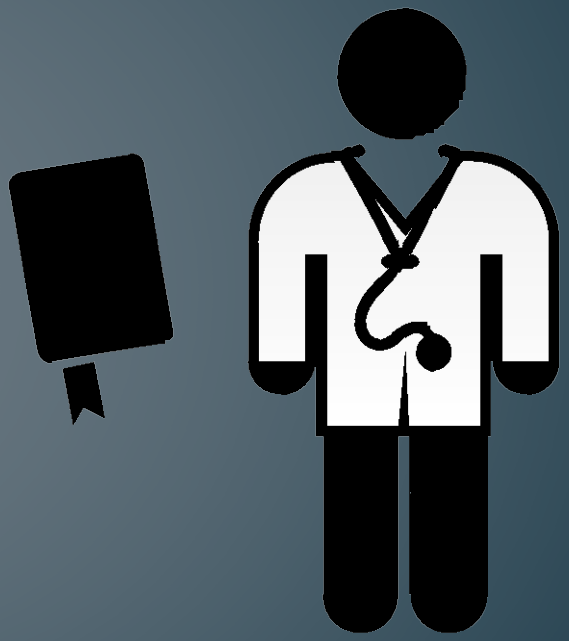
Petros

Dr. Ken

Palo Alto Medical Foundation

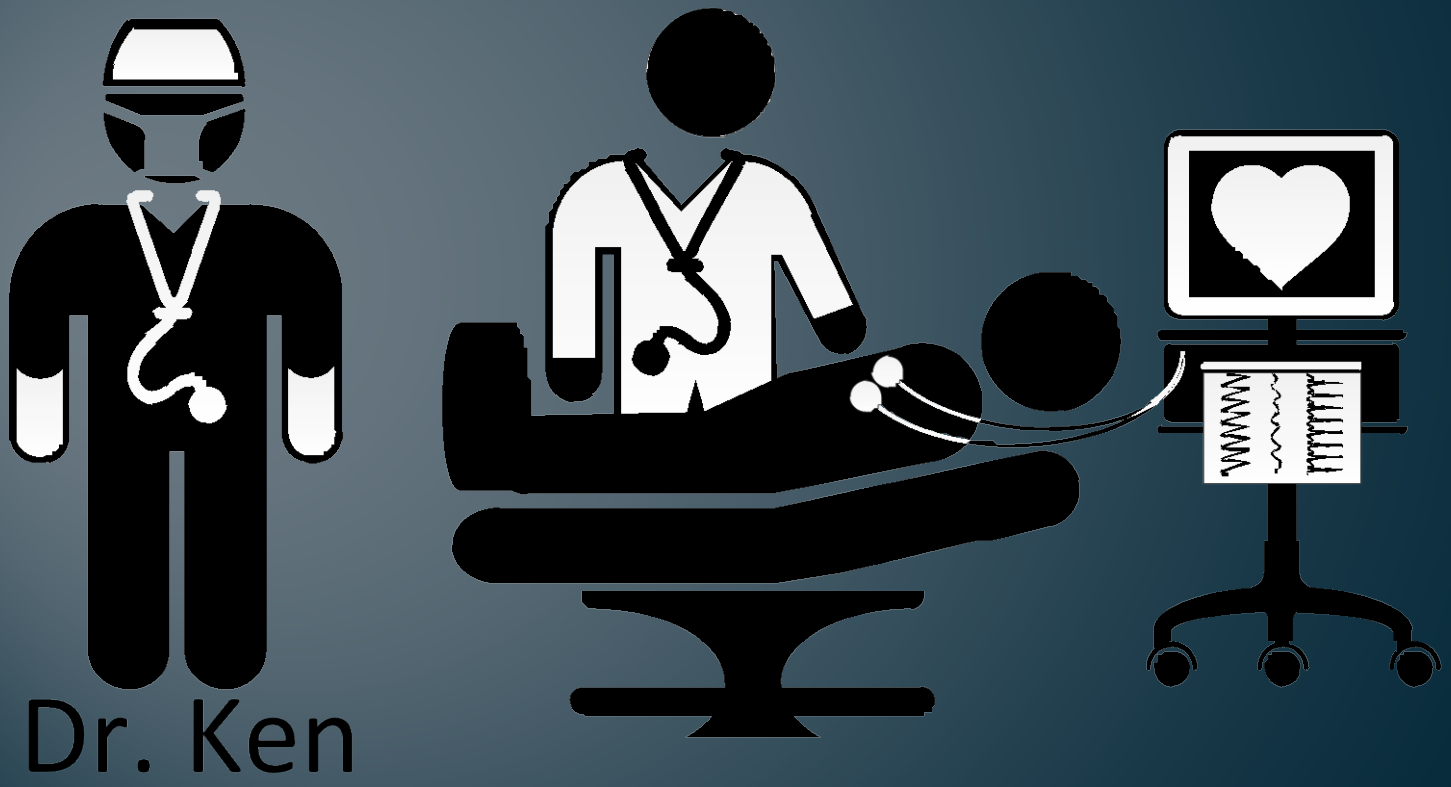


Palo Alto Medical Foundation

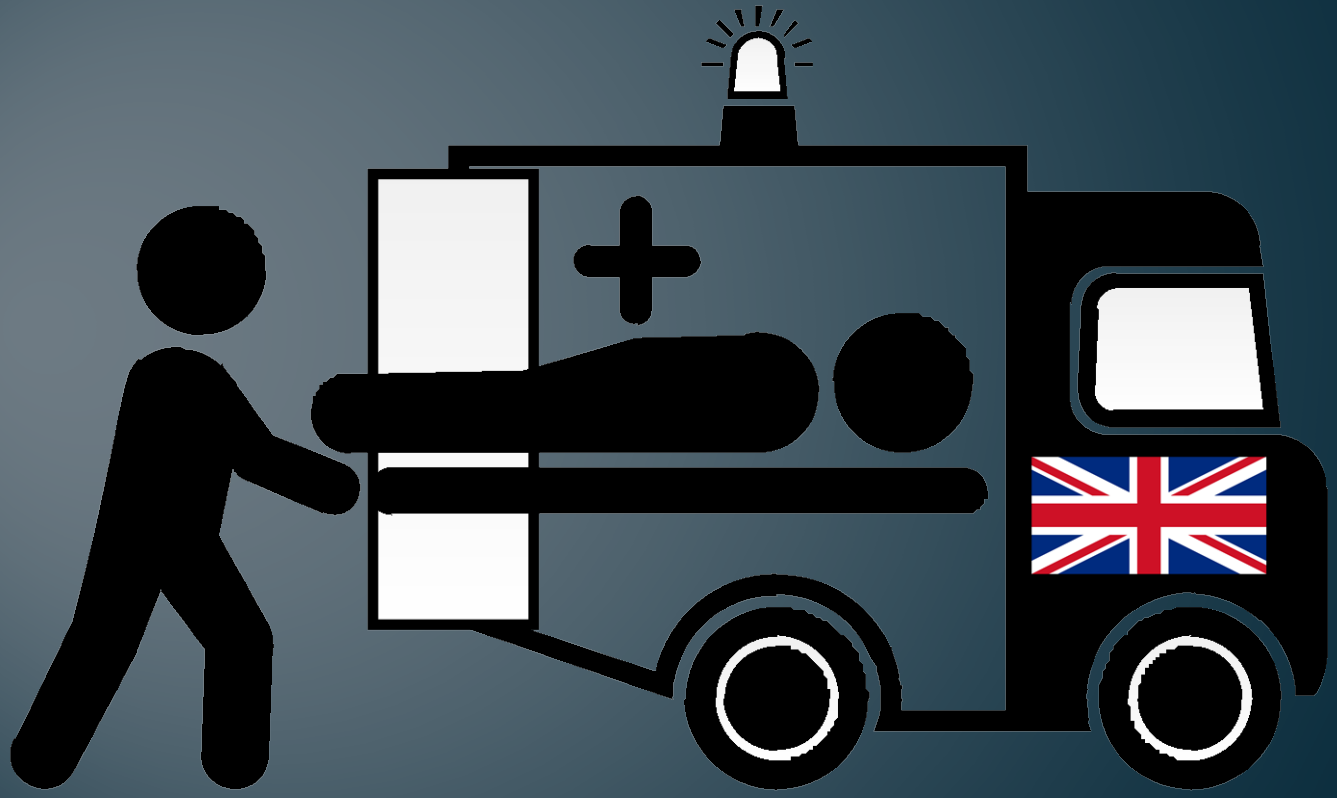


Dr. Magneto

Stanford Hospital



Swindon Hospital



Flexibility versus Data Protection

- Diverse modes of data use and storage outside owner's control
- Distinct organizations, infrastructures, jurisdictions
- Unknown software, maintenance, trustworthiness
- Deep, continuous, critical sharing

Today: trust everyone to do anything undetected, or die

Protect My Data In the Hands of Others

- Owner sets data policy: *Data Use Controls* (DUCs)
- Policy enforced on data while *out-of-custody*
- Data provenance maintained through all change

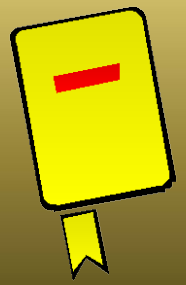
1. Support current OSes and applications without limiting choice
2. Remove OS, applications from the TCB, verify
3. Provide good performance

Legacy is the Killer App

Deriving a System Architecture

Unmodified OS

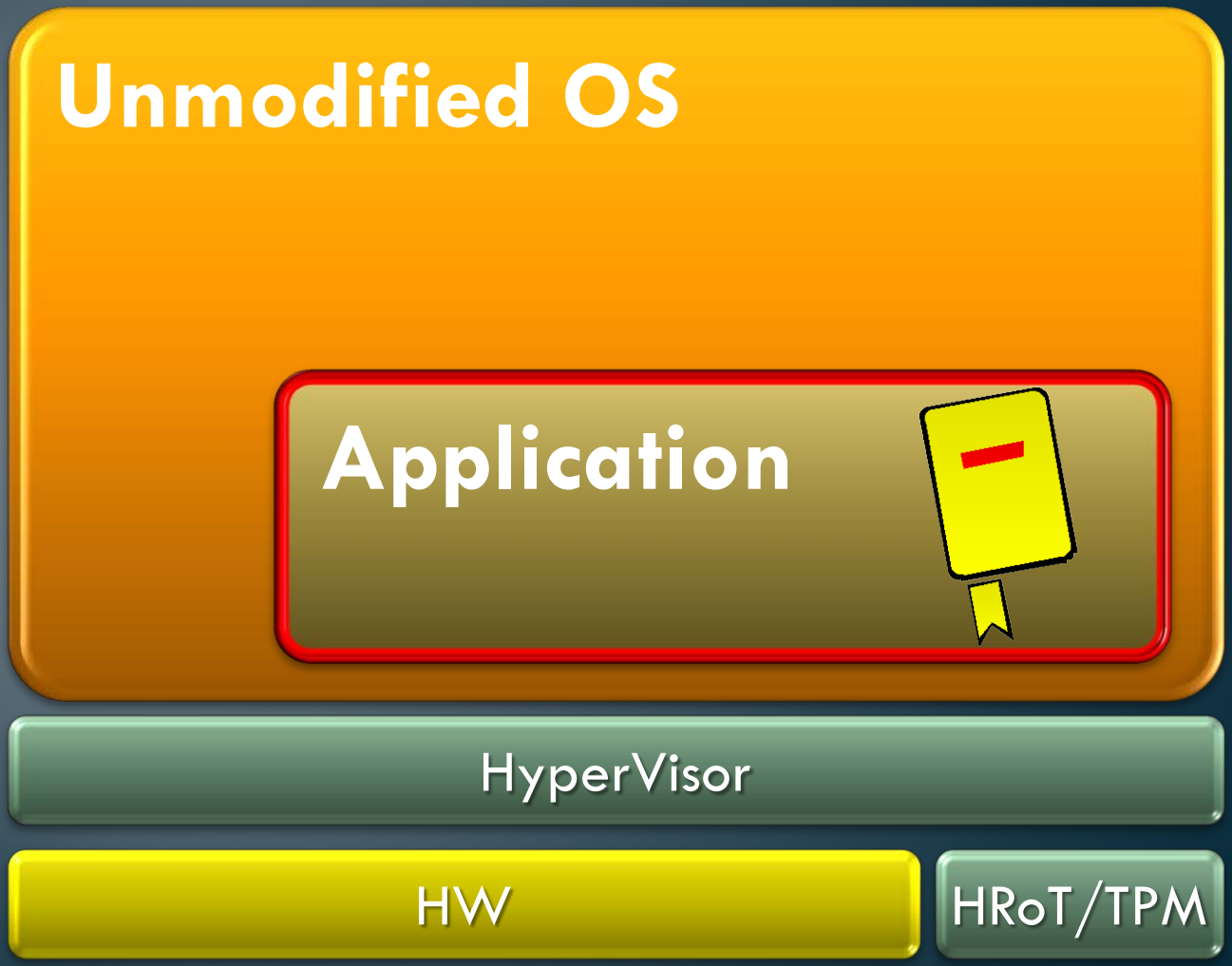
Application



HW

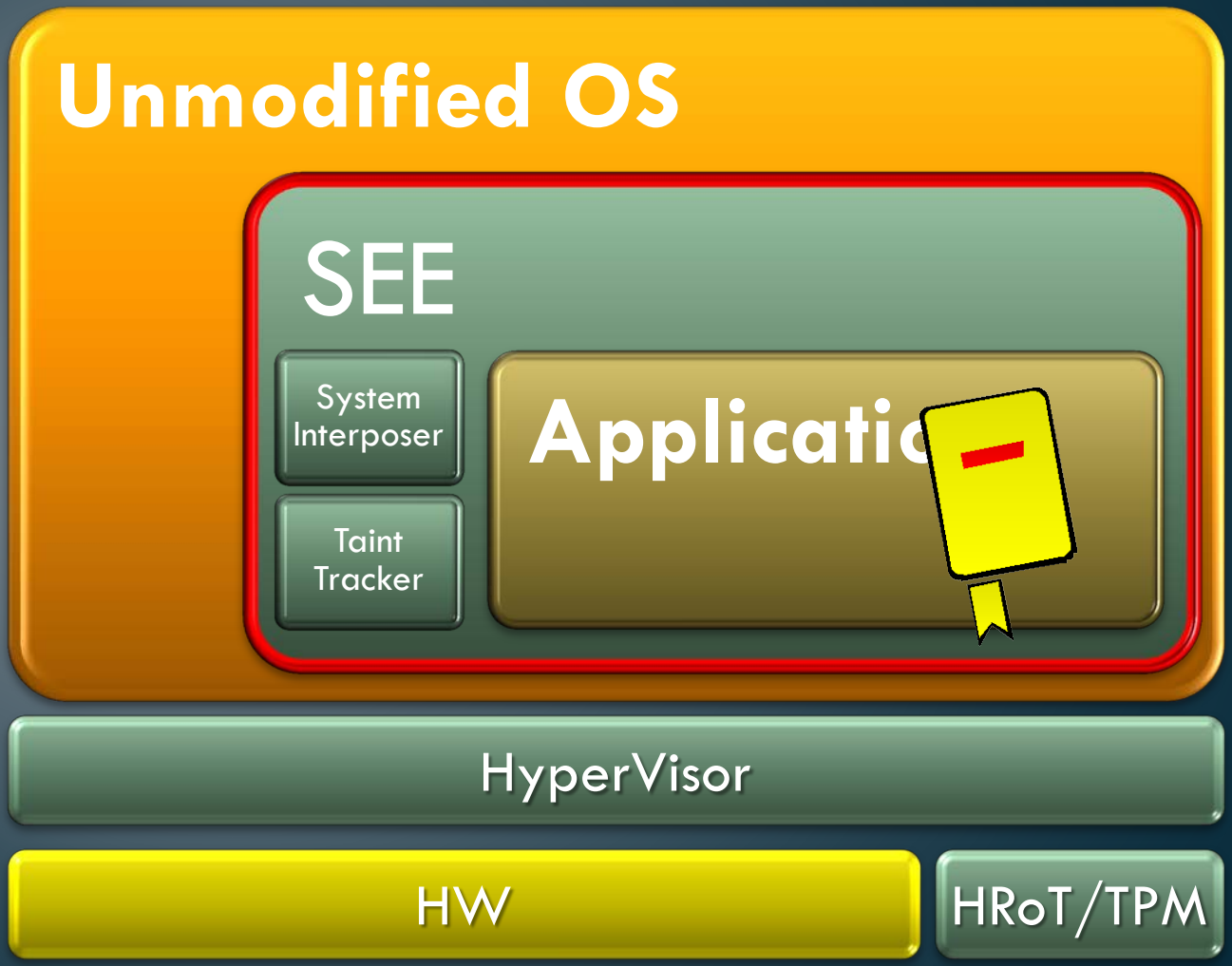
OS out of TCB →

App isolation from the OS + Remote attest



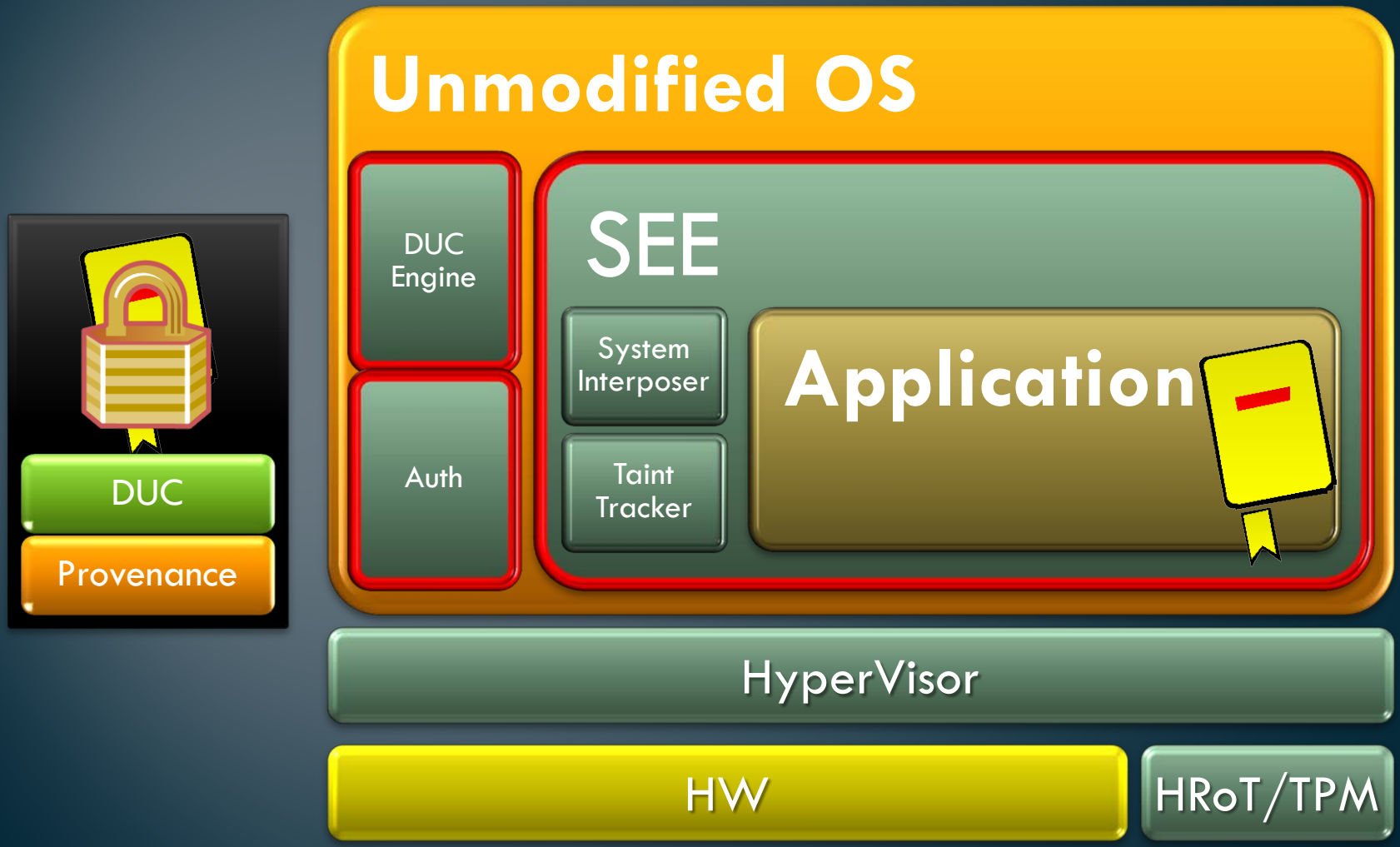
Application out of the TCB →

Sandboxing/Taint Tracking of Application



Limit impact on environment ➔

Encryption by default: Secure Data Capsules



The Life-cycle of a Secure Data Capsule

Palo Alto Medical Foundation

Petros's Foot MRI

- Mass here, mass there

DUC

- Dr. Magneto can edit
- Dr. Ken can view

Provenance

- Nurse Jackie Created

Palo Alto Medical Foundation

Petros's Foot MRI

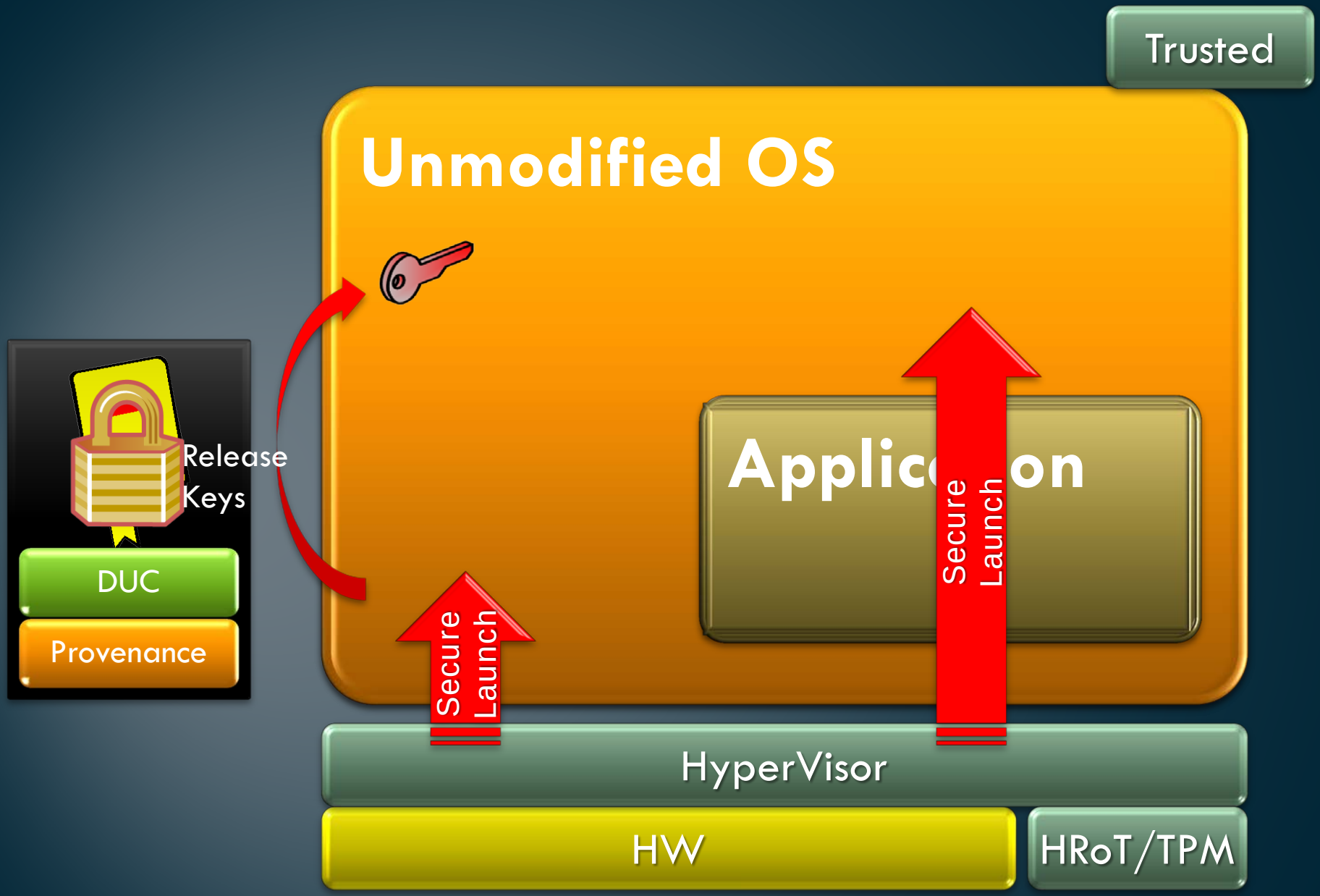
- Mass here, mass there

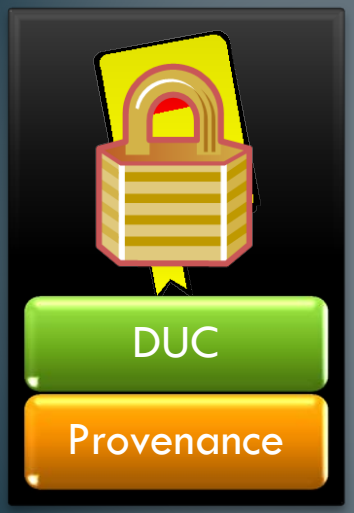
DUC

- Dr. Magneto can edit
- Dr. Ken can view

Provenance

- Nurse Jackie Created
- Dr. Magneto appended text, cropped image


Trusted
Unmodified OS
Release Keys
DUC
Provenance
Secure Launch
Secure Launch
HyperVisor
HW
HRoT/TPM



Trusted

Unmodified OS

DUC Engine

Auth

SEE

System Interposer

Taint Tracker

Application

HyperVisor

HW

HRoT/TPM

Trusted

Unmodified OS

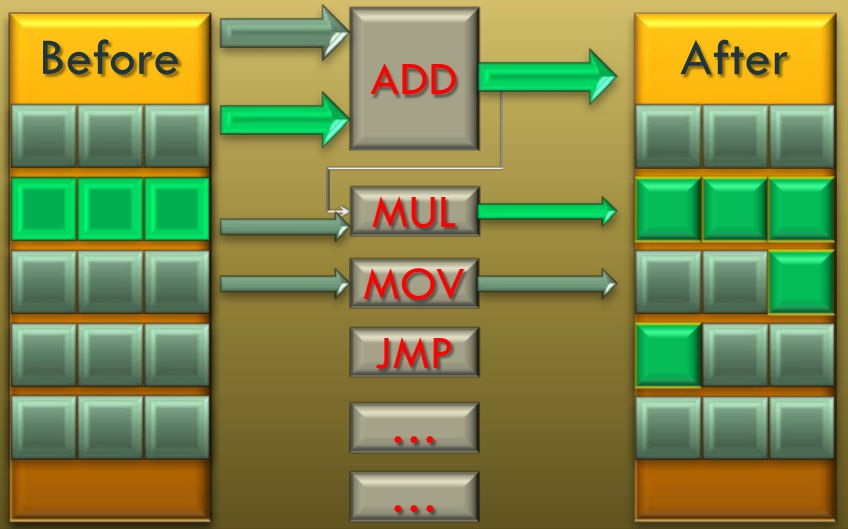
DUC Engine

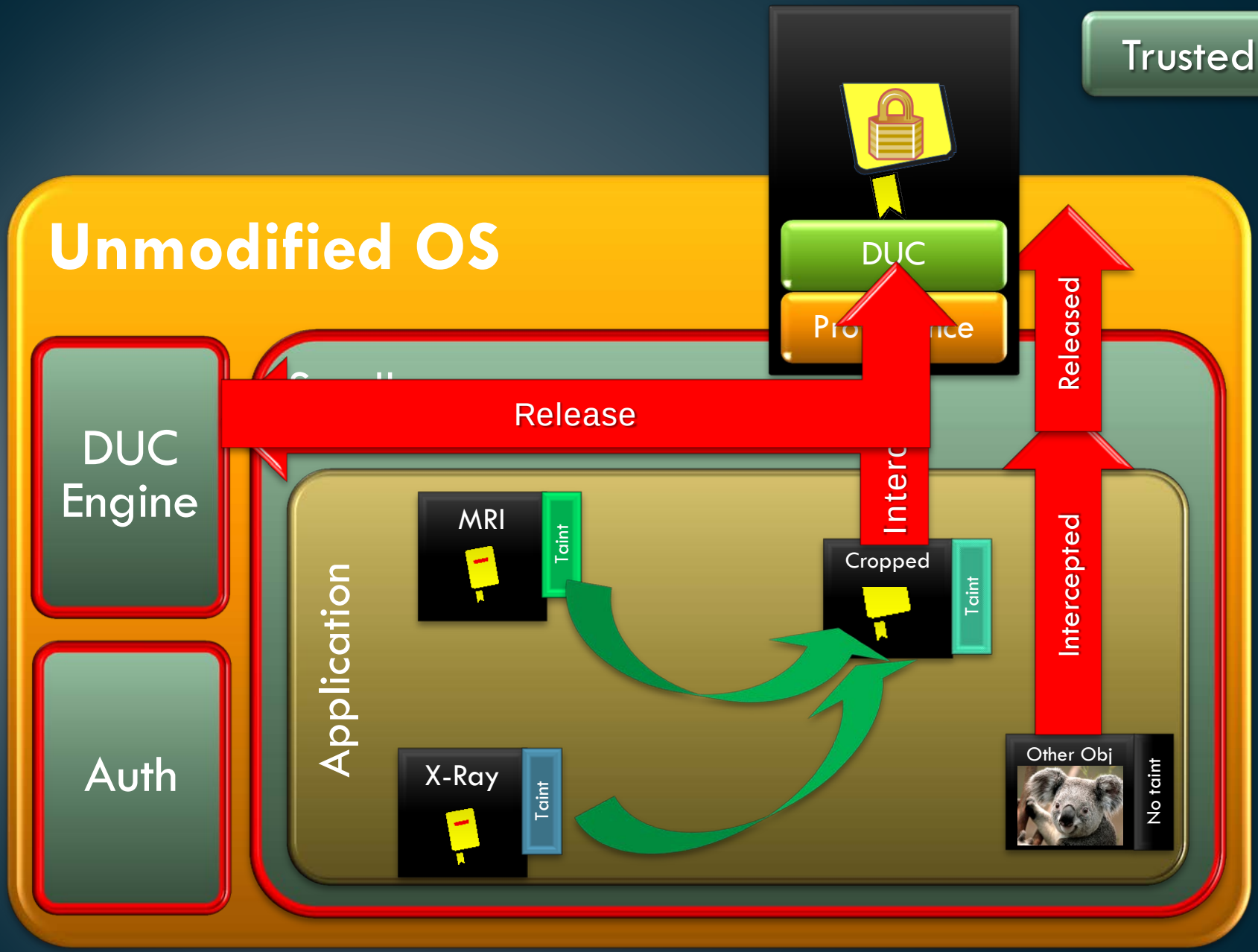
Auth

SEE

Tainted

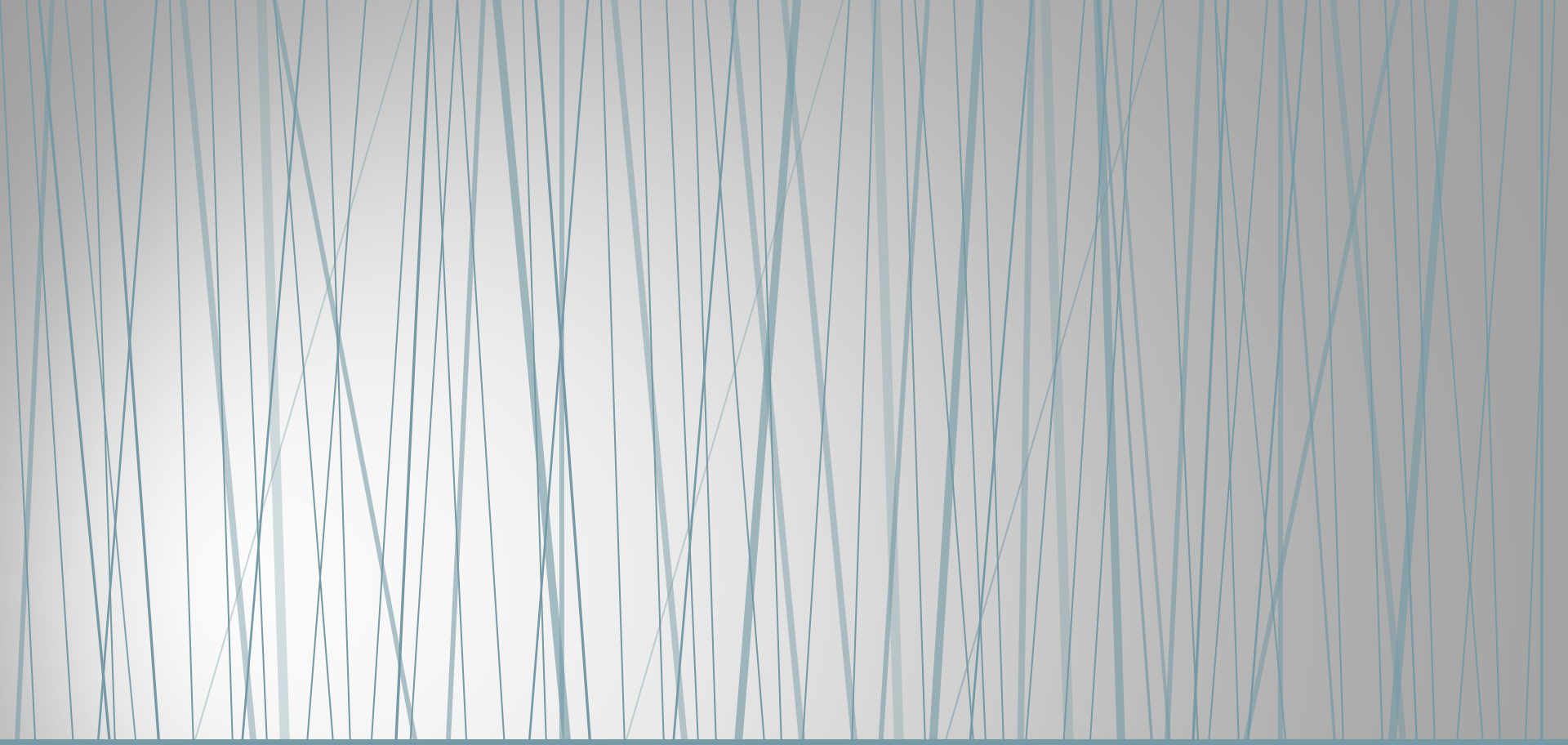
Application




Trusted
Unmodified OS
DUC
Pro ice
Release
Released
DUC Engine
Auth
Application
MRI
Taint
Cropped
Taint
Intercepted
X-Ray
Taint
Other Obj
No taint

Risks and Research Agenda

- Flow tracking does much of the heavy lifting, slooooowly
  - Might improve with: Restriction, Granularity, Asynchrony, Hardware
- How to keep as little as possible of policy evaluation and flow tracking in TCB? Why T, CB? Prove it, please!
- Are DUCs meaningful to humans? Composable? App-specific?
- Covert channels a serious threat with untrusted applications
  - A tussle between flexibility – leak-ability, what can we do in between?
- Aggregation/analytics?



Q&A

Thank You!

Saturated Nomenclature: Capsules

- Ampoules
- Caplets
- Flasks
- Pods
- Cocoons
- Sheaths
- Husks
- Bob

What Doesn't Solve The Problem?

- The Enterprise Rights Management approach
  - Everyone uses same SW platform, applications
  - Like begging for non-compliance
  - Tough across organization/jurisdiction boundaries
- Decentralized Information Flow Control
  - New OSes: small TCB but incompatible (e.g., HiStar), or compatible but large TCB (e.g., Flume)
  - New languages (e.g., Jif): rewrite applications, no protection at OS custody
  - Red/Green models: Trust application, disallow sharing, coarse granularity

Design Alternatives

- Sandboxes can have variable semantic richness

1. Know nothing of semantics
   - Act as a data sink, block all output except display
   - Storage Capsules [Borders2008]
2. Understand information flow
   - Allow output of data, sharing across apps
   - Must track flow of sensitive bits to outputs (DIFT)
3. Understand application or data semantics
   - Need trusted enforcers for app-specific policy

- For now, targeting #2 with support for #3