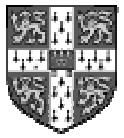


UNIVERSITY OF CAMBRIDGE
Computer Laboratory

HSMs and Security APIs: Enabling Trusted Computing

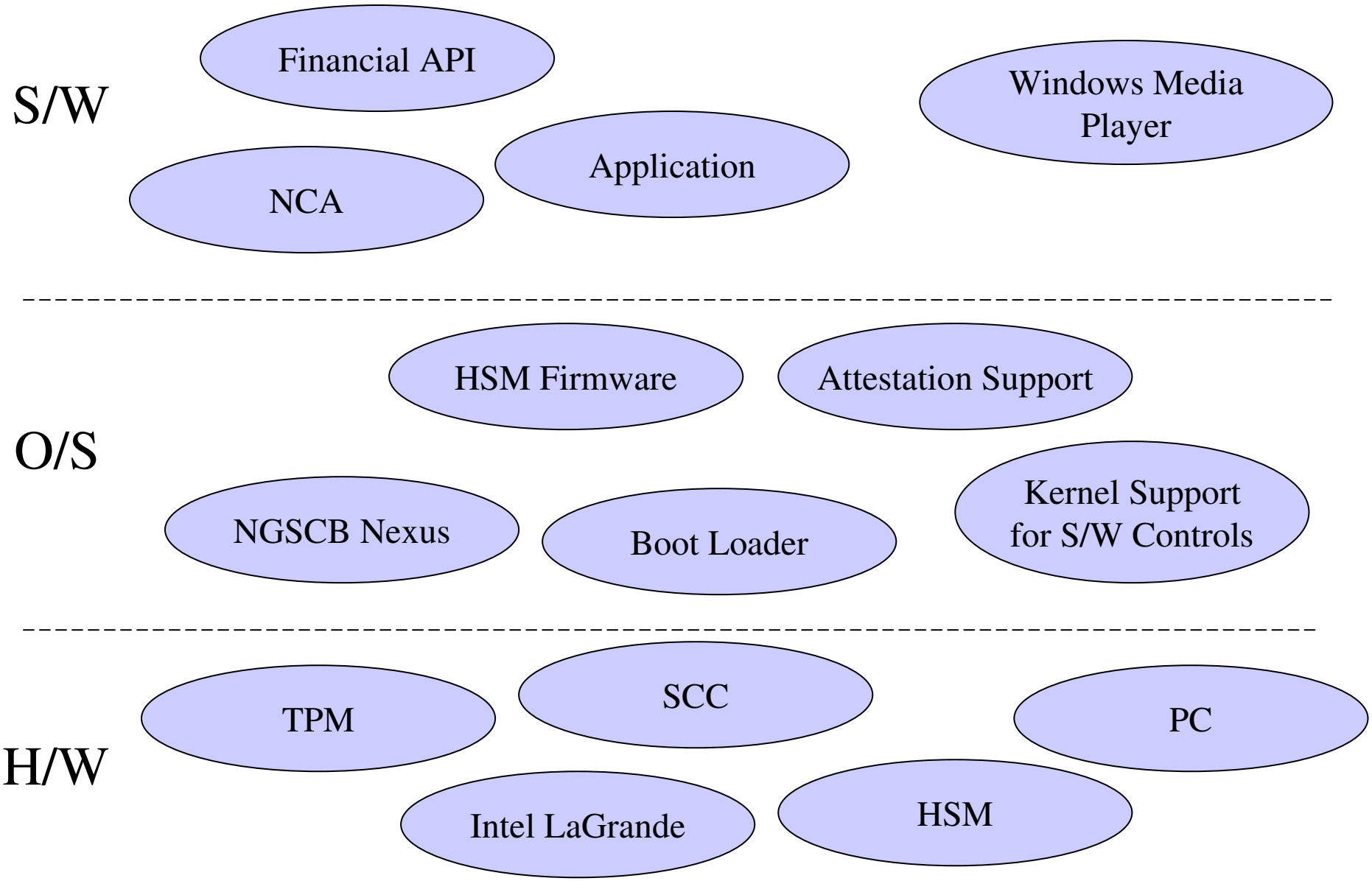
Mike Bond
Computer Security Group

Information Security Solutions Europe **30th September '04**

Talk Structure

- Introduction: The Trusted Computing Stack
- TC Hardware
- TC Operating Systems
- TC Software
- Introducing Security APIs
- Application Level Security
  - The Identification Challenge
  - Version Control and Updating
  - Secure Backup
  - Helper Applications
- Conclusions

The Trusted Computing Stack



Talk Focus

- ***How do we design and build our Trusted Computing applications?***
- Where do we start?
  - Trusted Computing Specs are sets of moving targets
  - TCG Hardware (TPM) fairly well defined now, but this is too far down the stack to have all the answers for the application designer
  - MS NGSCB: not even the overall architecture has been agreed on
- Proposal: can we learn from applications already designed and deployed on the original TC platforms: Hardware Security Modules?

In Brief : TC Hardware

- A number of paradigms
  - The Hardware Security Module
    (The original island of trust. Many will run arbitrary code)
  - The Passive Trusted PC (e.g. TPM-based)
    (Monitors state, but does not enforce – all or nothing control)
  - The Active Trusted PC
    (e.g. Intel LaGrande, MS SCC, Trusted Graphics & Sound Cards)
  - The Proprietary Box
    (Microsoft XBOX 3? Sony PS3? Apple iThingy? Centre of your home area network, completely closed protocols)

Hardware Security Modules

- Specialised computer hardware (e.g. PCI cards) for implementing Security APIs and performing cryptography.
- Characteristics:
  - Often physically tamper-resistant
  - May have hardware crypto acceleration
  - May have special ‘trusted’ peripherals
- Limitation: Unlike potential for TC, HSMs mainly run a *single application* at a time
  e.g. PKCS#11 **or** PIN Processing API.
- HSMs are not “vapourware” like some TC, they are out there, designed and deployed.

Hardware Security Modules

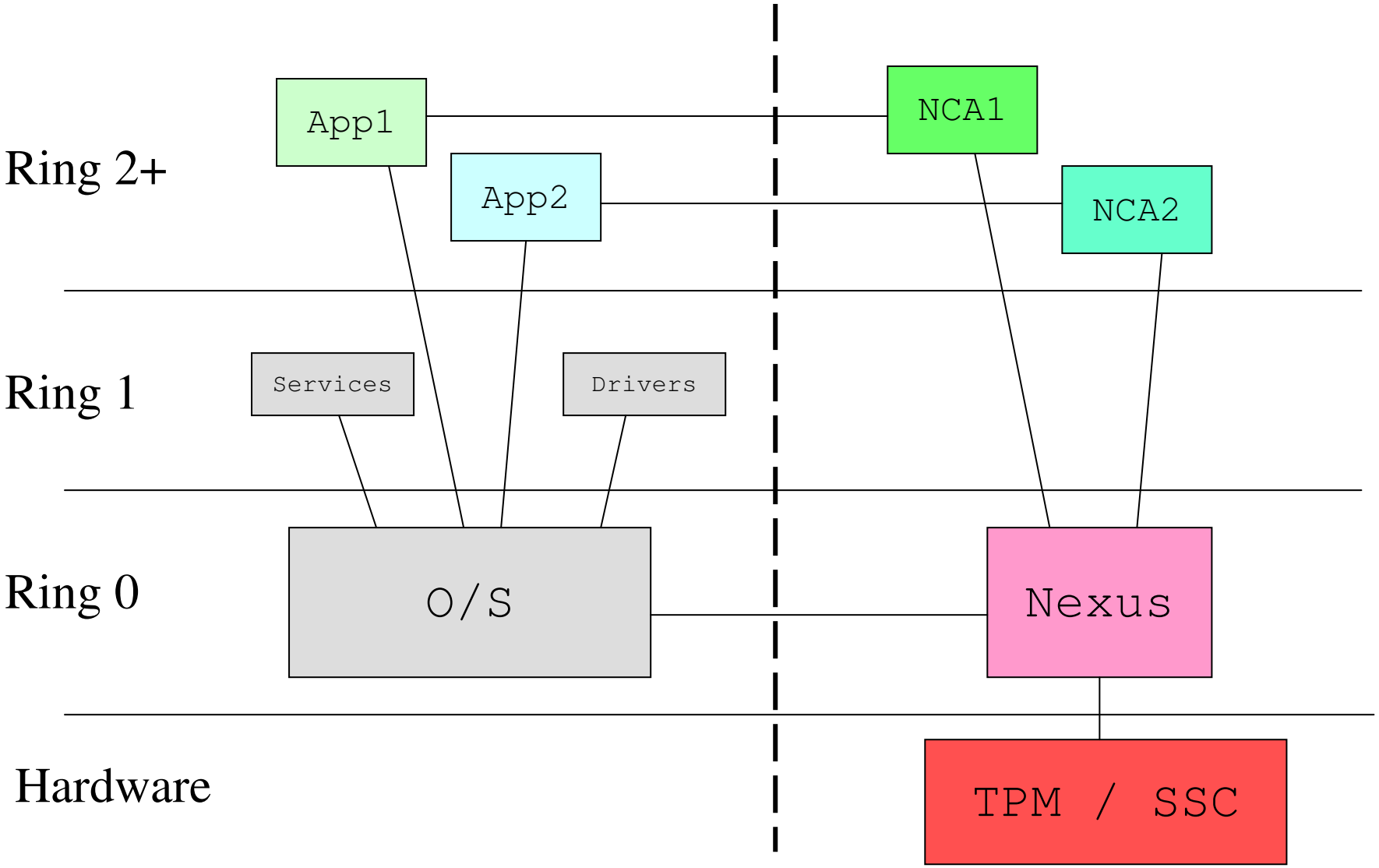
Enabling TC With HSMs

- Using HSMs
  - TC applications involving licensing, or content distribution have many secrets to protect
  - TPM/LaGrande based TC not currently highly tamper-resistant, we need a more secure place for our top-level keys and business logic – answer: use HSMs for top levels of hierarchy.
  - HSMs already standard for top levels in CAs.
- Learning from HSMs
  - HSMs have been designed and built since the mid 80s. TC can learn a lot from understanding their hardware, OS and application design.

In Brief : TC Operating Systems

- People have tried to build trustworthy O/Ses for years, there was limited success: main problem was security/usability compromise
- Microsoft trying to build trust into operating systems (Palladium, NGSCB). Their idea: to retain compatibility, keep existing OS controls, and branch off in a new axis too. Will it make it in time for Longhorn?
- Plans for Linux with TCG technology
- We are also seeing DRM-Aware Operating Systems in mobile phones

Example: Microsoft NGSCB



(More on this later…)

Focus : TC Software

- What goes at the top of the stack?
- We already have **heavy-weight** applications we might wish to TC enable
  - Office/Lotus Notes (Information Rights Management)
  - IC Design Packages (Protect proprietary routing algorithms)
  - Windows Media Player (Protect digital content, new charging strategies)
- Some applications already use software obfuscation to achieve same goals – can we harden these applications?
- What new applications can we build?

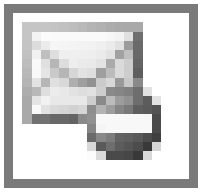
TC Applications (Good & Bad)

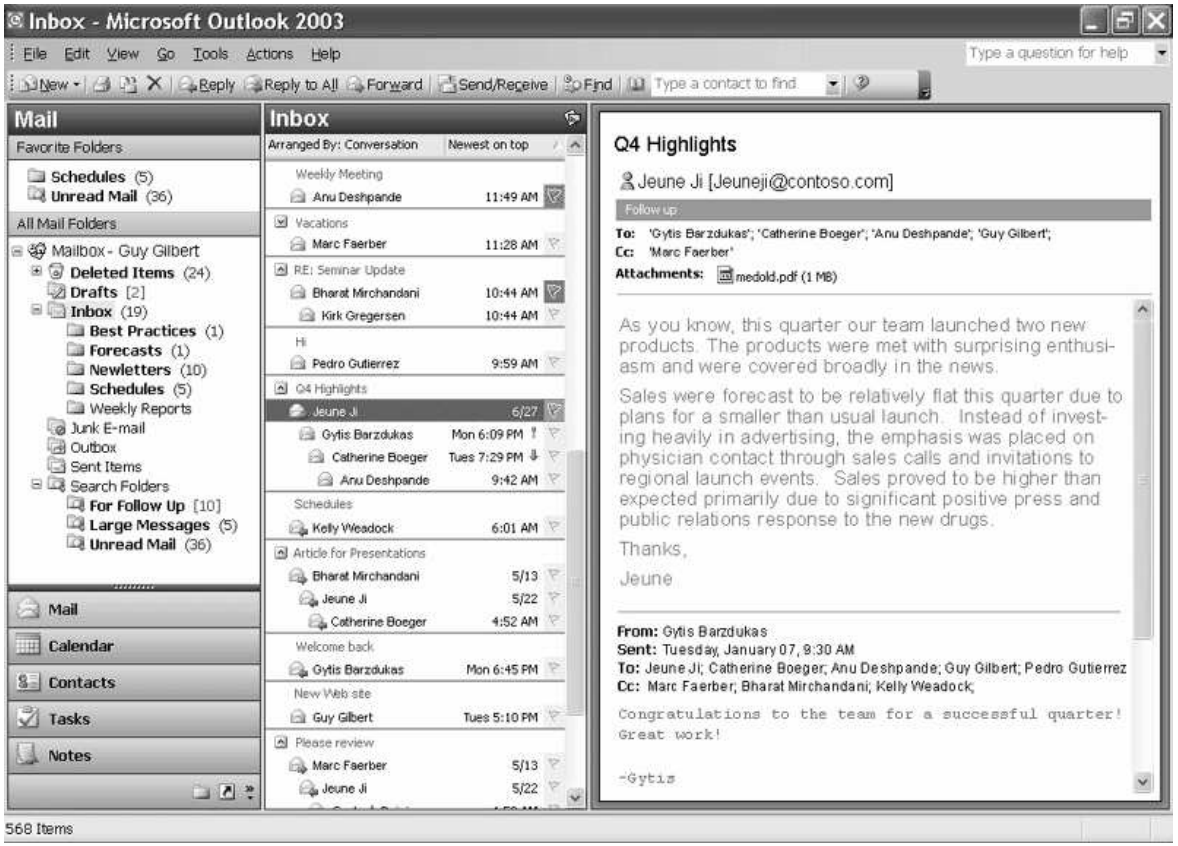
- IRM – Information Rights Management
  - Companies can stop leaks
  - Mafia can keep their records secret
- DRM – Digital Rights Management
- Proprietary Algorithm Protection
- Trusted I/O – Enter your ATM PIN at your PC
- Global PKI – All devices potentially indentifiable
- Trusted Anonymity Systems
- Truly Anonymous peer-to-peer systems
- High-availability systems
- Reverse-engineering resistant viruses

Example: Information Rights Management

- Microsoft Office 2003 with
  Microsoft Rights Management Server
- Will it be secure when it becomes TC enabled?

The “restrict” button



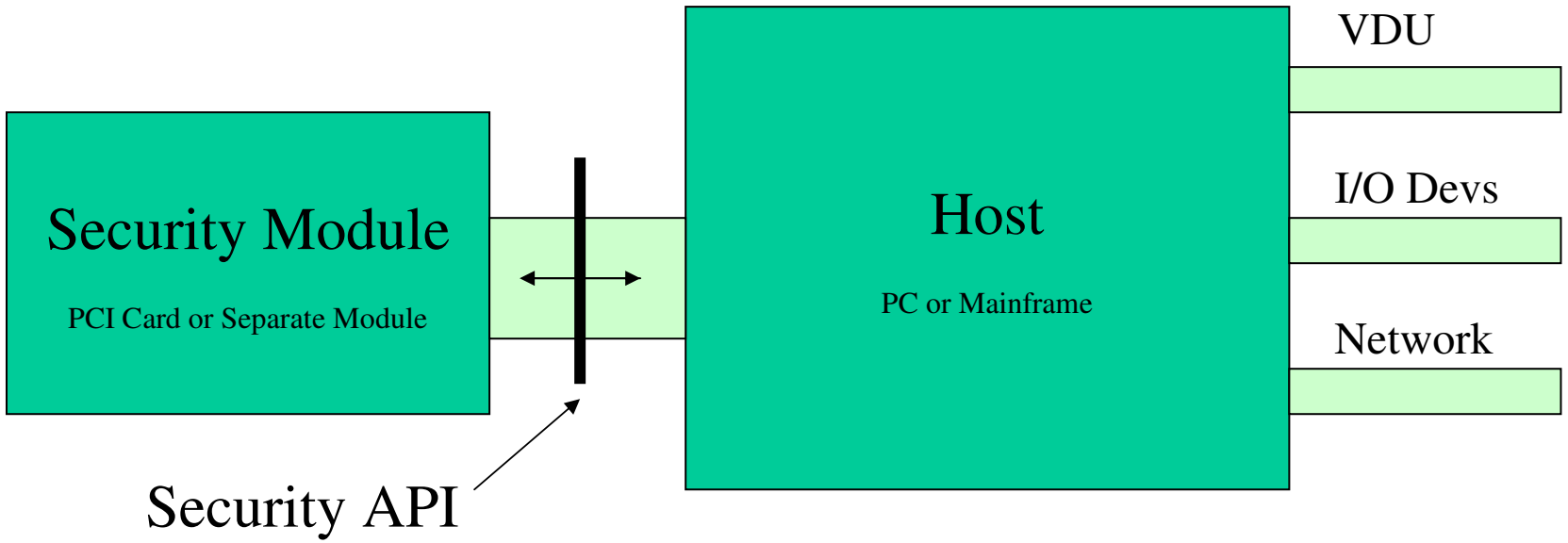


The Trust Boundary Problem

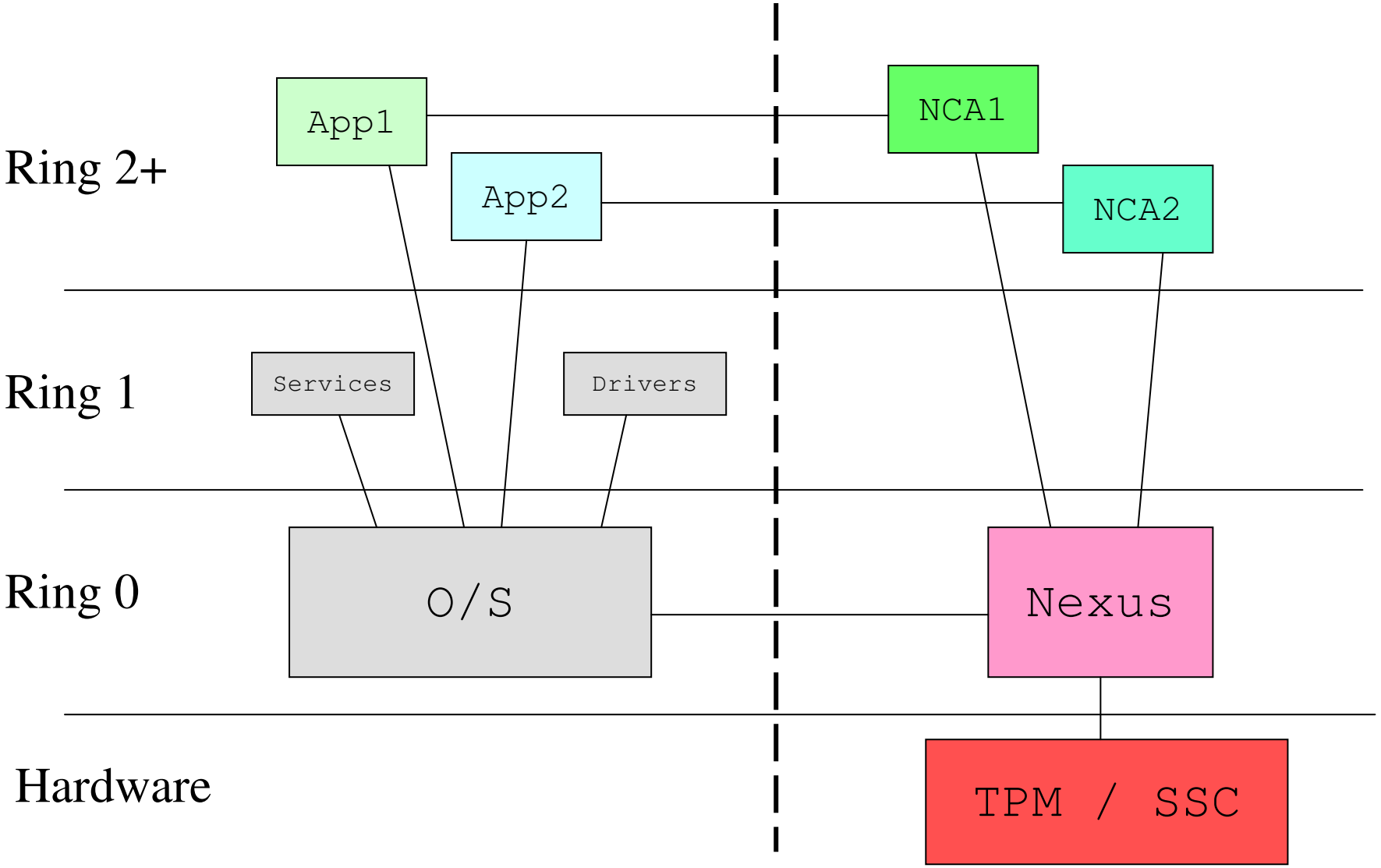
- The crucial question: which parts of an application go inside the trust boundary?
- We can't make a whole monolithic app trusted (otherwise it will be full of buffer overruns etc.).
- If too little is trusted, then only very simple policies will be enforceable (no fancy new payment models)
- This problem has been thought about by trusted app designers using HSMs, but unfortunately remains unsolved. There has been a whole catalogue of failures of HSM APIs. Security API research is the place to start looking for the answers, so that TC designers at least do not repeat the same mistakes…

What is a Security API ?

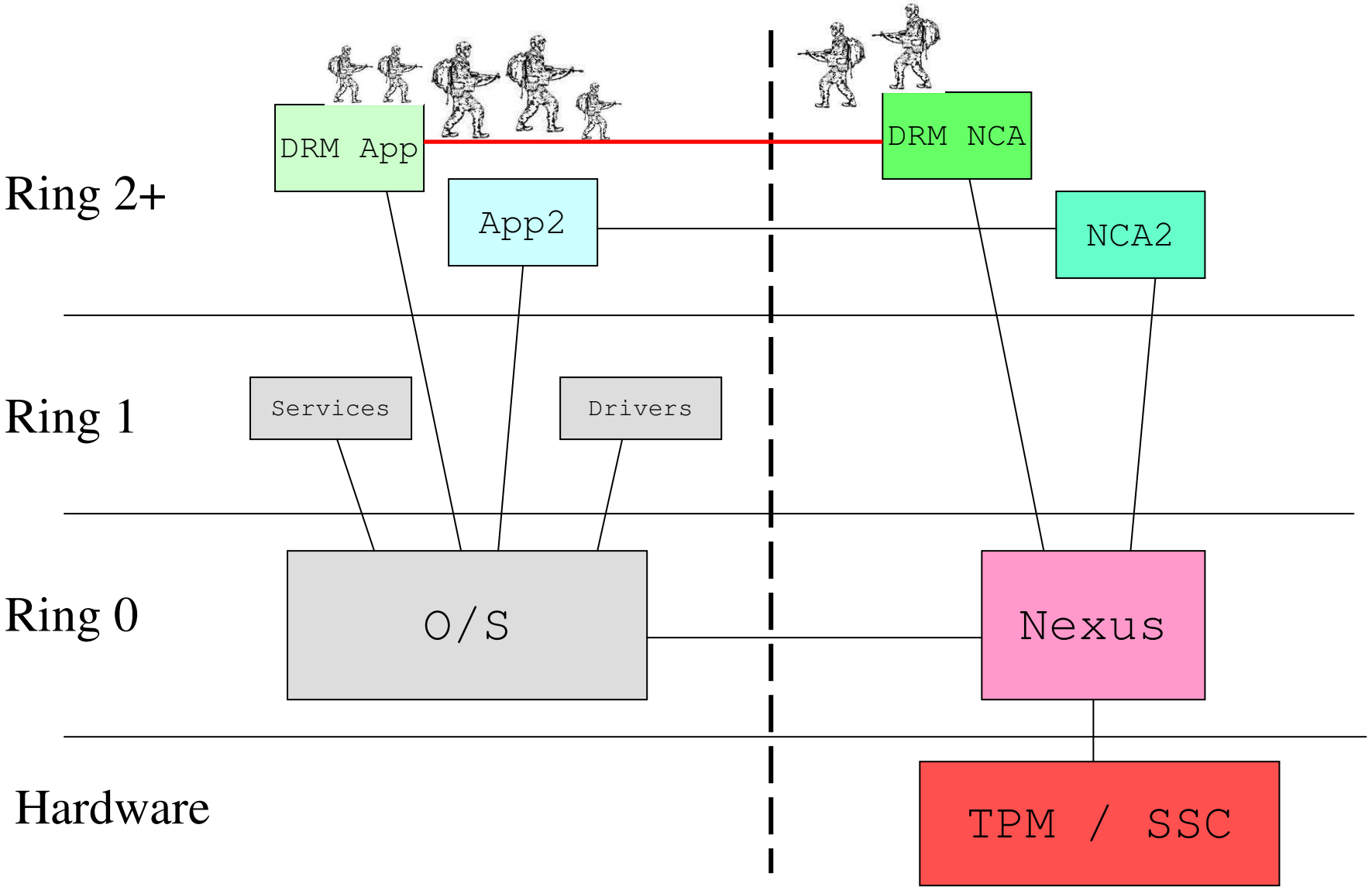
- An API that allows users to work with sensitive data and keys, and uses cryptography to enforce a policy on the usage of data
- The *conventional* Security API:



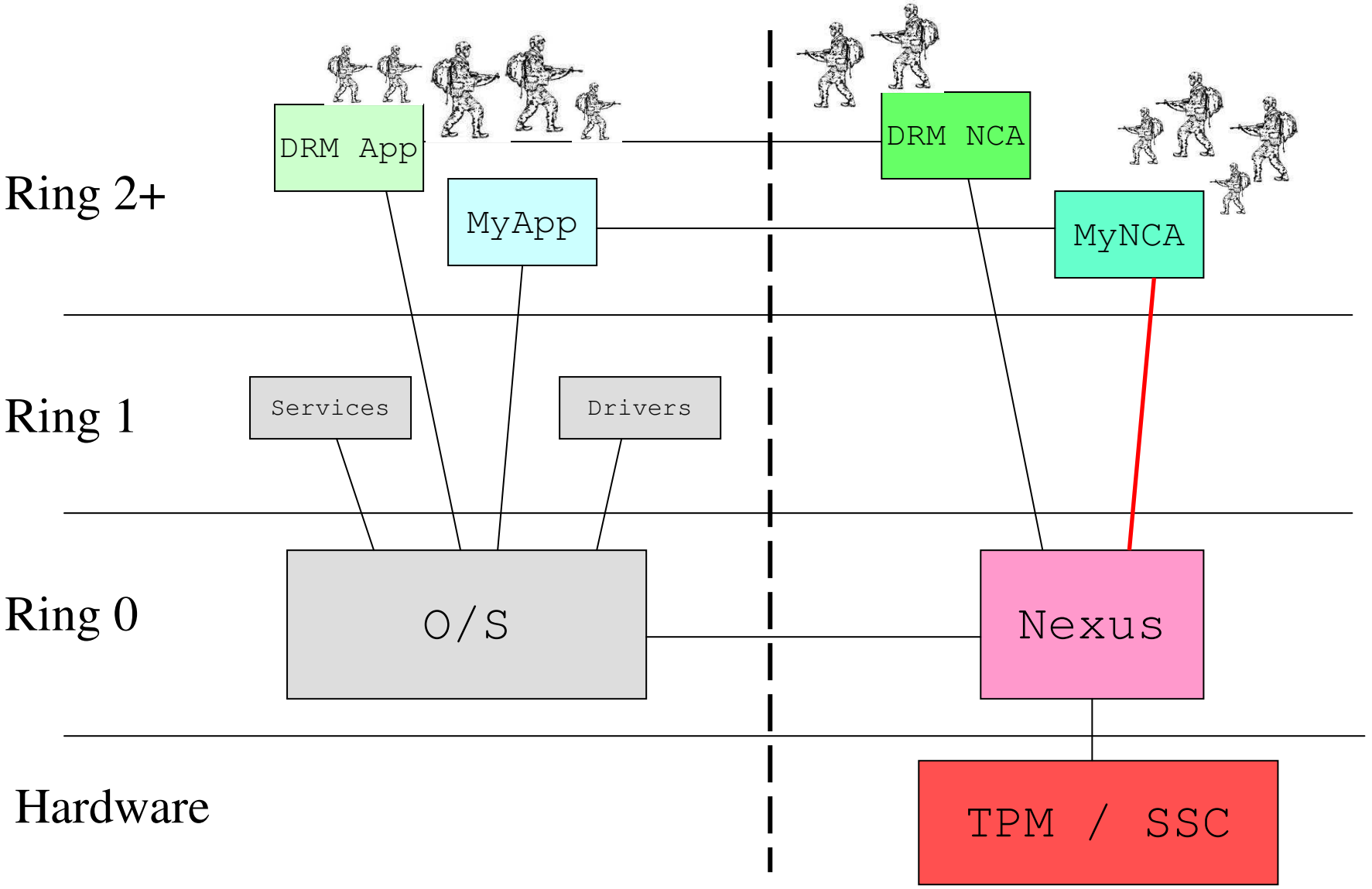
TC Security APIs



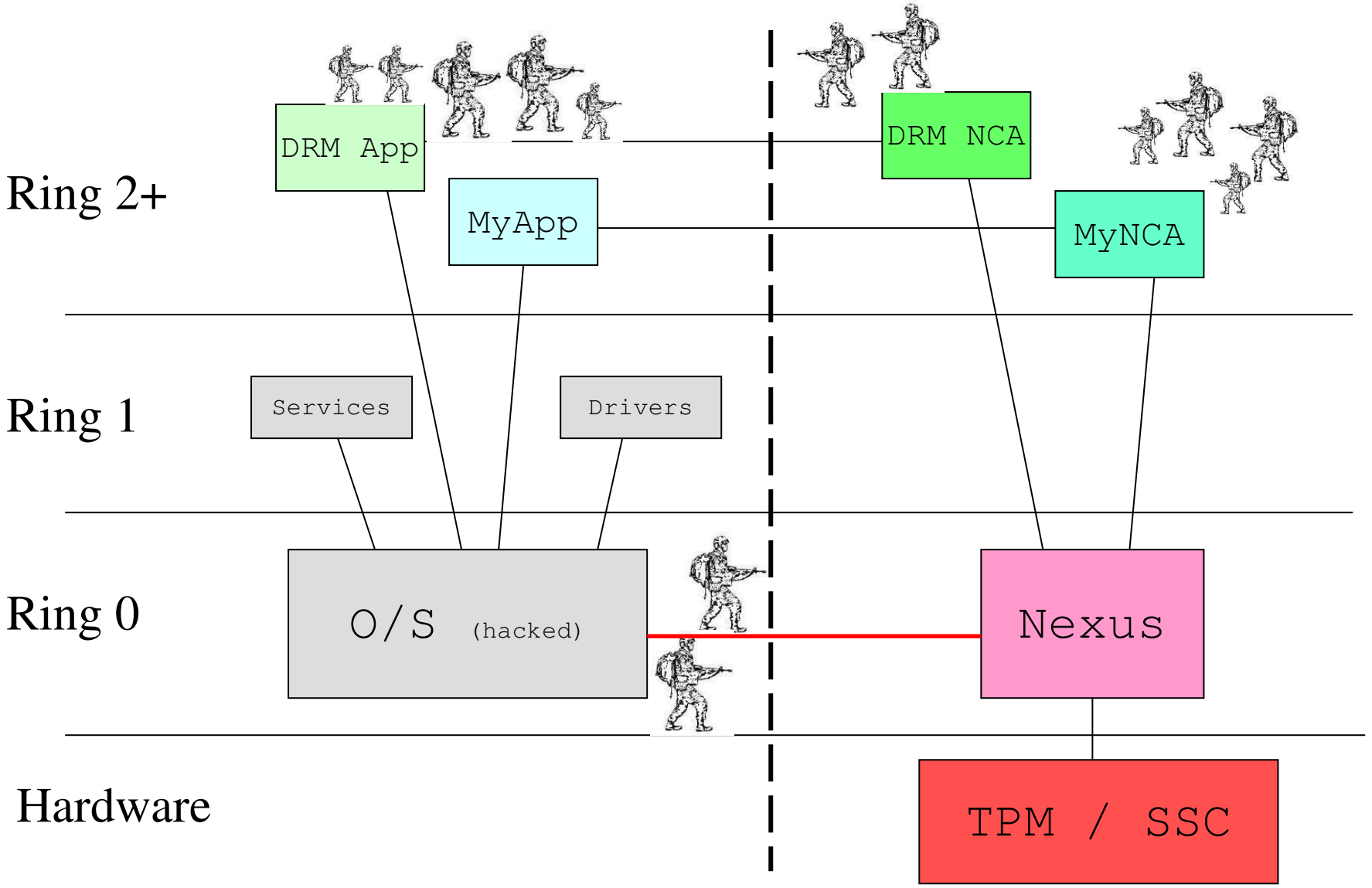
TC Security APIs



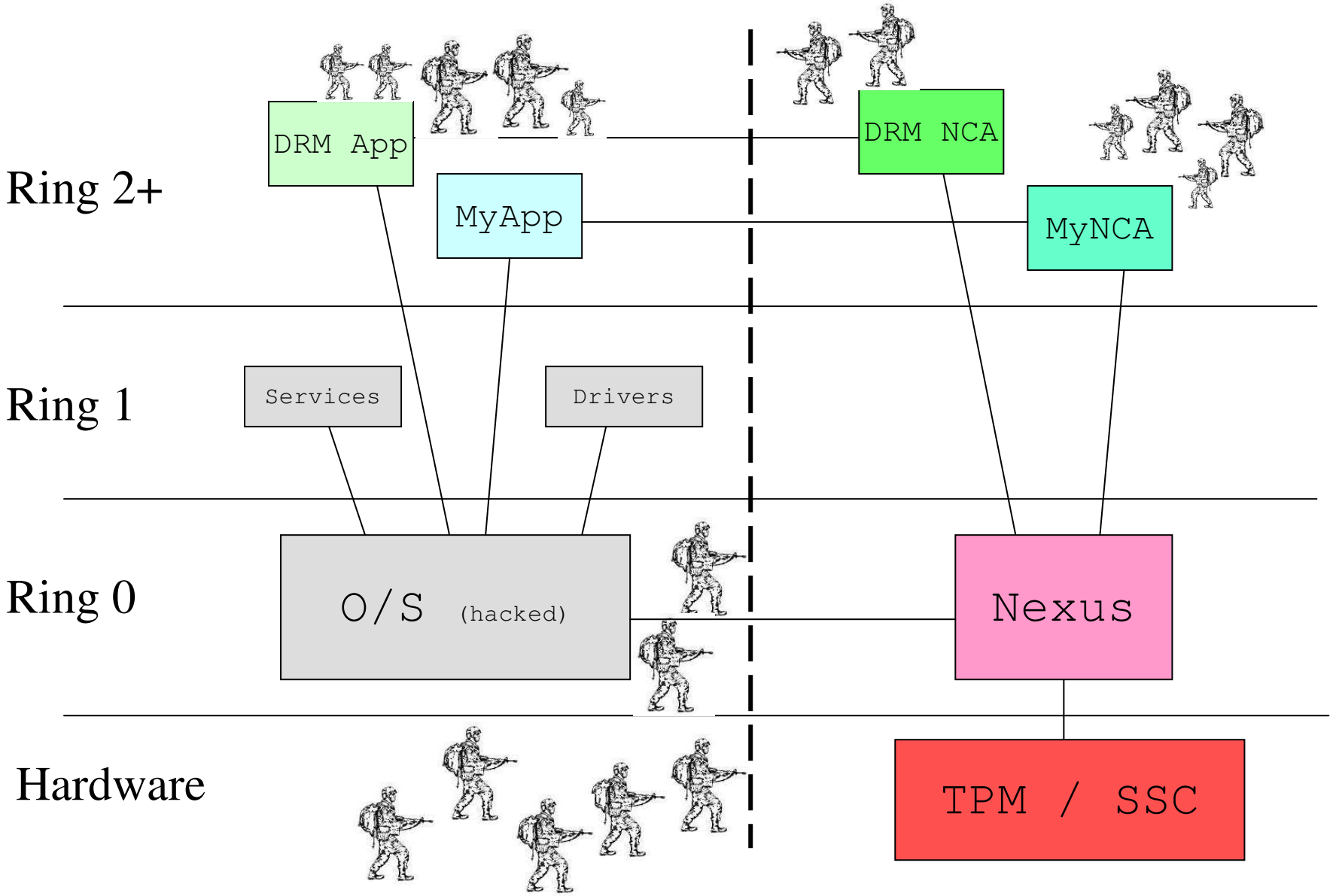
TC Security APIs



TC Security APIs



TC Security APIs



Security API Successes & Failures

- Financial APIs for processing PINs. A very complex heterogeneous system: many different boxes at many banks. A stream of vulnerabilities discovered:
  - Key Binding, Weak 3DES (2001)
  - Information Leakage, Decimalisation Table Attack (2003)
  - ISO PIN Block Attacks (2004)
- Depending on your point of view, PIN processing systems are either ***too hard*** to get right, or ***too expensive*** to get right.
- Certification Authorities. Simple policy: “the private key never leaves the box”. On the whole, a success.
- The better we understand the problem, the cheaper it will be to build quality solutions

Application-Level Security

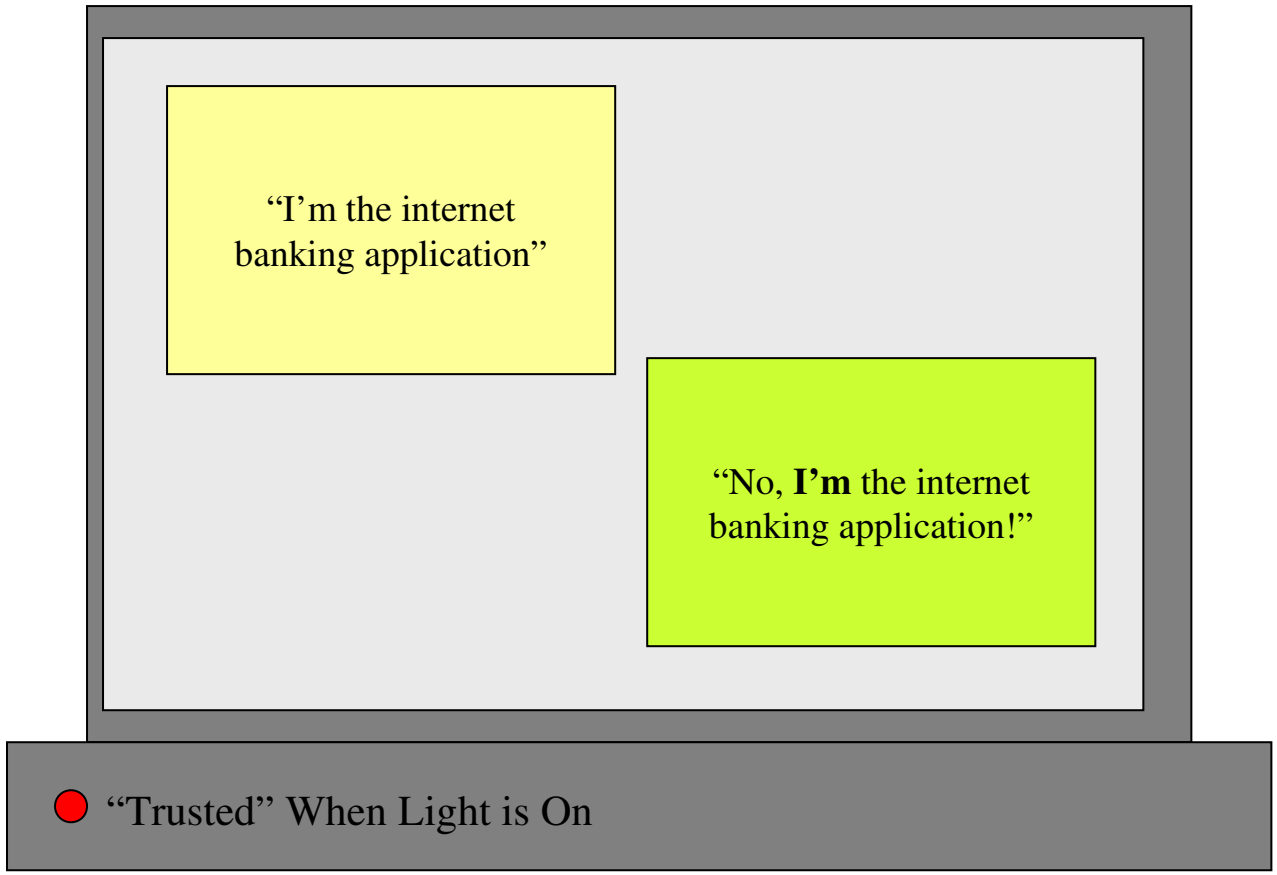
- Some application-level security challenges have already been tackled by today's Security APIs
  - Major vendor APIs support straightforward back-up, access control, auditing, and specific business logic
- However, all these problems get more complex when they are generalised. Modern Security APIs can often solve the challenges single-application scenarios, with infrequent update, but the solutions used become impractical …
- Let's consider how these issues develop.

The Identification Challenge

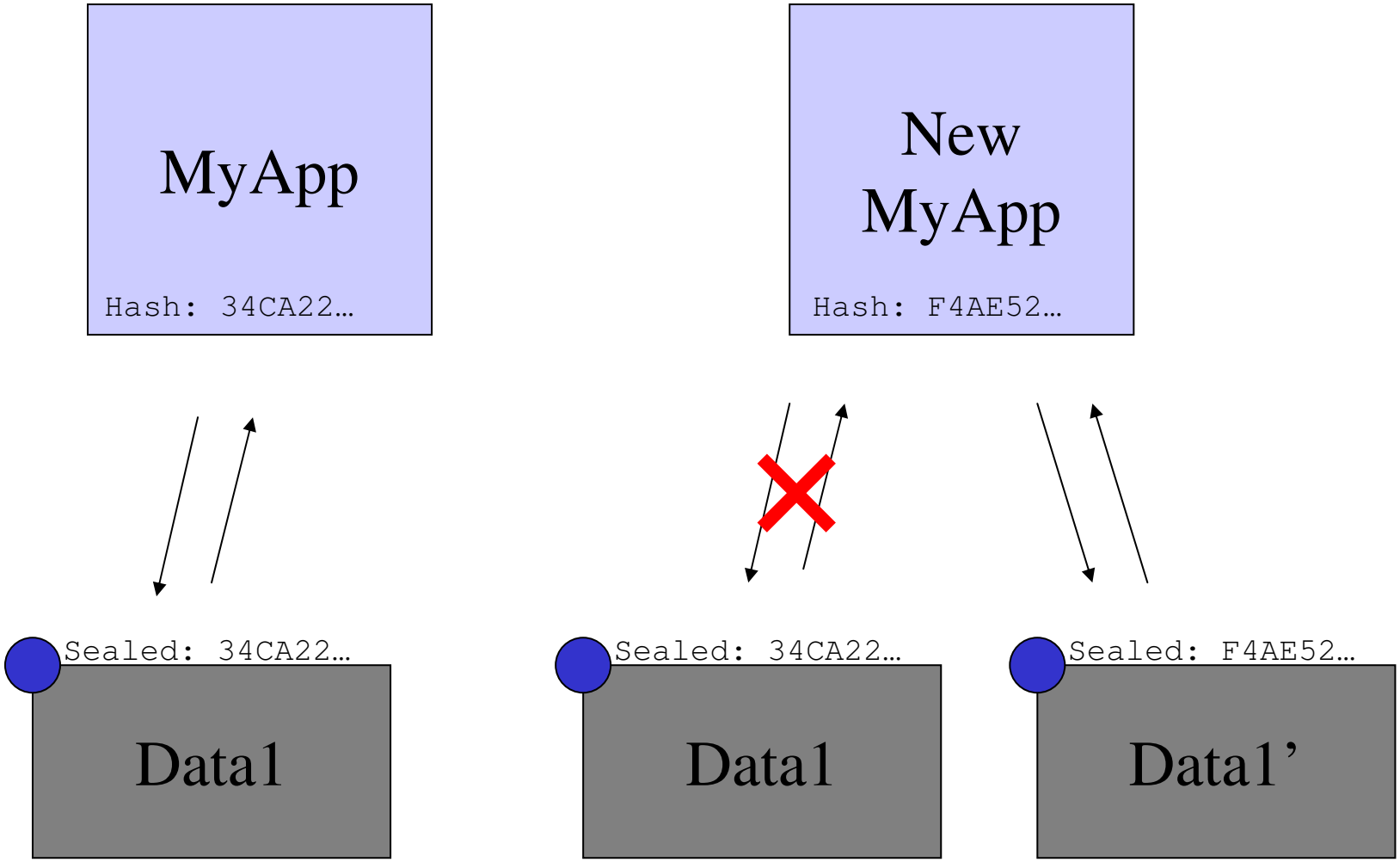
- Many HSMs recognise that I/O through the host computer can be sabotaged to be misleading, so have special authorisation and feedback paths:
  - nCipher uses a smartcard insertion to enable e.g. a limited number of signatures with a private key
  - Thales RG modules have a special “console” connection for issuing of sensitive commands
- But what about when there are myriads of different trusted apps under different ownership?
- Trojan apps can be easily developed (e.g. phishing scams), and if the user has not seen a particular application before, it is trivial to make a plausible imitation

The Identification Challenge

- How do you know **which** trusted app you're talking to?



Version Control and Update



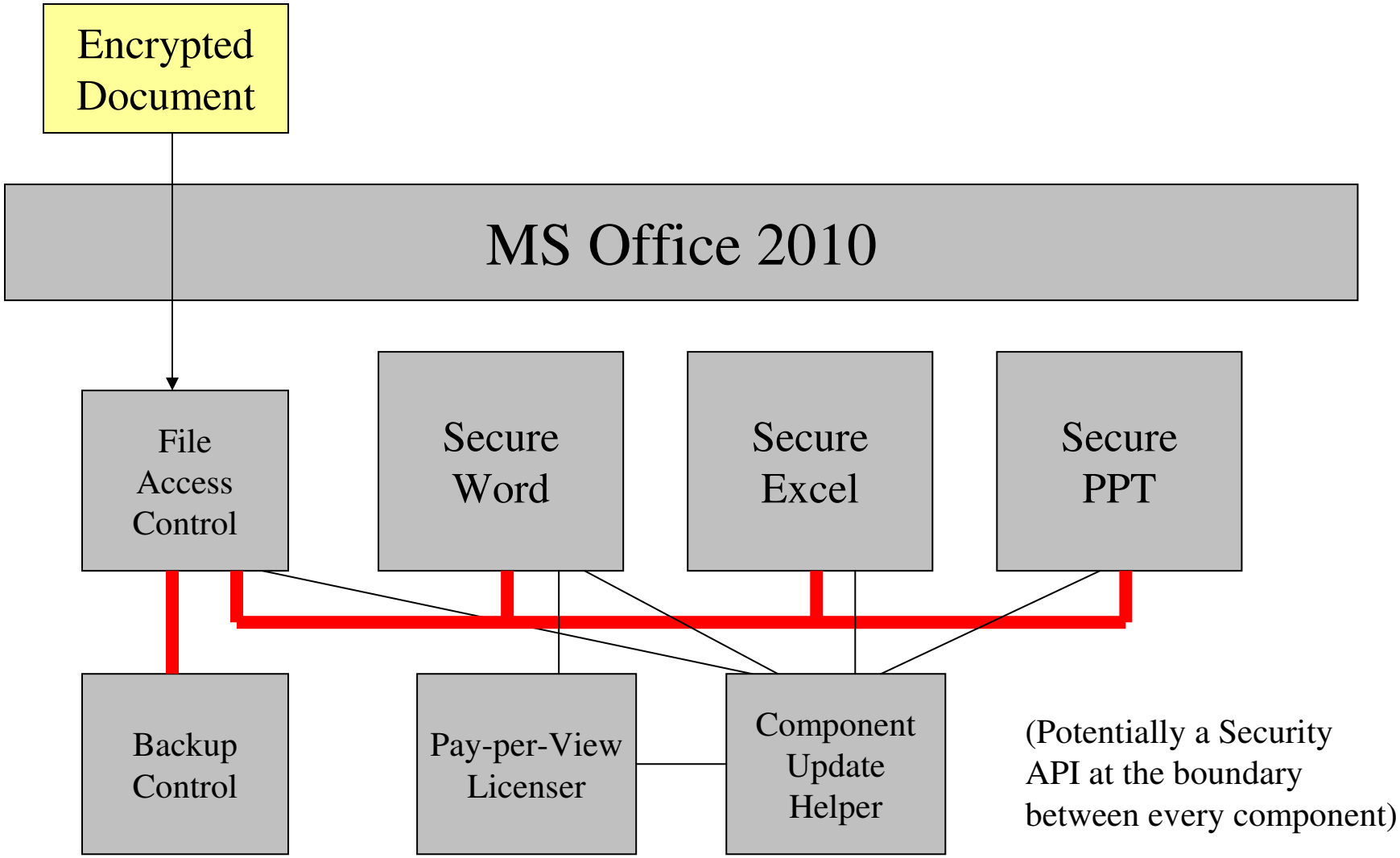
Version Control and Update

- Toy Problem before, the solutions seems straightforward – give the old program a migration routine to re-seal all the files to the new version.
- Real world complexity
  - How to migrate offline backups?
  - What about un-installation and roll-back? (suppose the new software has an unrelated bug that crashes the client PC all the time)
  - What if you're trying to patch a fault in the migration routine?
  - How often will you have to migrate? Every time a DLL the app uses changes? Every time you add a new plug-in to an app?
  - What do you migrate? Suppose the PC has multiple users, running different versions, or one uses Windows Media Player for jukebox, one uses RealPlayer – who owns which music files? What about shared music files?

Secure Backup

- Users have to trust that they will always be able to access their data, and retrieve it from back-up if necessary
- What if the user buys a new PC? What are the back-ups bound to, the user, the application or the PC?
- What if the user restores back-ups onto a different PC, conflict with DRM use-limits?
- What if a manufacturer goes bust?

Solutions: Helper Applications?



Conclusions

- We are making good progress laying down base architecture for TC, but there's still more work to be done. We must not neglect the design of applications that sit on top, and must not assume that the correct O/S support for TC will make it easy to write trusted apps.
- Studying the Security APIs of applications built on HSMs can give us a good starting point for TC application design.
- However, there are some fundamentally hard problems raised when mutually distrustful parties share computing resources. We need to work very hard on these to make a success of desktop Trusted Computing.

More Information

Papers, Links & Resources

http://www.cl.cam.ac.uk/~mkb23/research.html

Cambridge Computer Laboratory

http://www.cl.cam.ac.uk/Research/Security/

Colleagues and Related Work

http://www.cl.cam.ac.uk/~mkb23/phantom/

http://www.cl.cam.ac.uk/~jc407/

http://www.cl.cam.ac.uk/~rja14/

Email... `Mike.Bond@cl.cam.ac.uk`