seL4 Summit 2019

seL4 Drivers for Trustworthy Devices:
towards capabilities from silicon thru apps

stu.card@critical.com

Alice and the Red Queen running hand in hand
illustration by John Tenniel, 1871, now Public Domain
https://commons.wikimedia.org/wiki/File:Alice_queen2.jpg

This work was performed as CTI IR&D plus SBIR/STTR efforts with Syracuse University, Genode Labs & Data61, supported by DARPA, AFRL/RI, AFOSR & OSD. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the DoD or the U.S. Gov't. Cleared by DARPA as DISTAR case 29869.

user -> developer -> researcher story with intentionally ambiguous title ☺

- "devices"
- "trustworthy"
- "drivers": broad issues
- motivating example applications
- prototype gateway stack components
- Genode / seL4 verification
- Syracuse Assured Boot Loader Executive
- "drivers": device driver software
- "drivers": lessons learned
- next steps

our primary objective as / on behalf of users: return meaningful control of devices and networks to their proper owners, by enabling a safe exit from the Red Queen's race

“trustworthy”

- not just trusted: a decision to trust a system should be based on evidence
- several dimensions
  - **Reliable**: doesn’t fail in isolation
  - **Scalable**: doesn’t fail just because many are interacting
  - **Secure**: doesn’t fail due to malicious action
  - **Resilient**:
    - continues to provide some level of service even if partly broken
    - facilitates rapid restoration of full services
- not binary: varying degrees of confidence along each dimension

“drivers”: broad issues / aspects

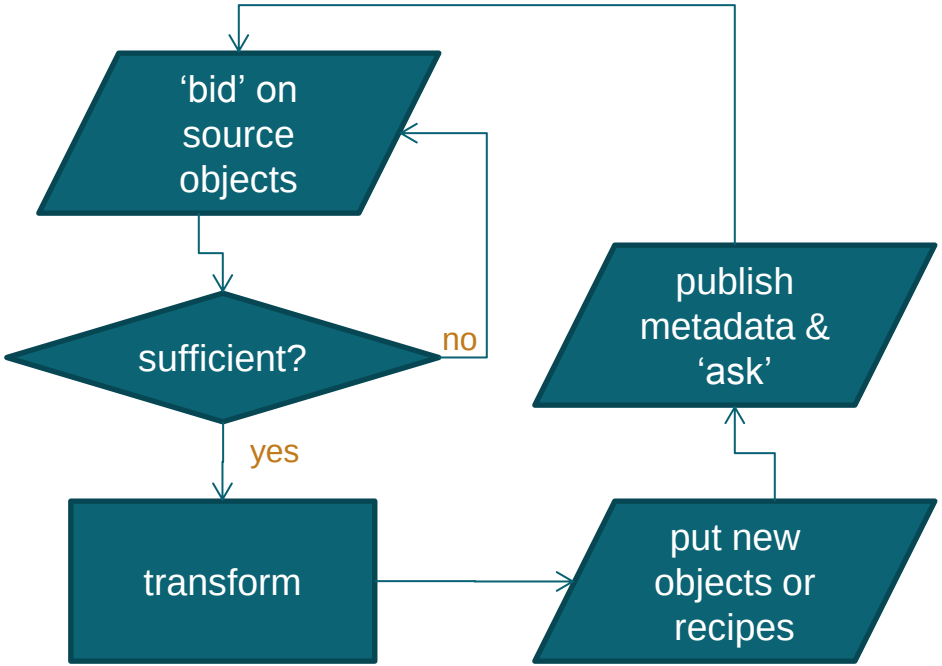
- System
- Protocol
- Software
- Hardware
- Energy/Power: not the microkernel’s problem directly, but e.g. an energy-draining attack depends on the scheduler to get the time to drain the battery
- User Training: if a user trusts the system to do A, but it has actually been designed [& verified] to do B…

motivating example applications

(we have really worked on 3 of these for clients, 3 as IR&D, and are proposing the rest)

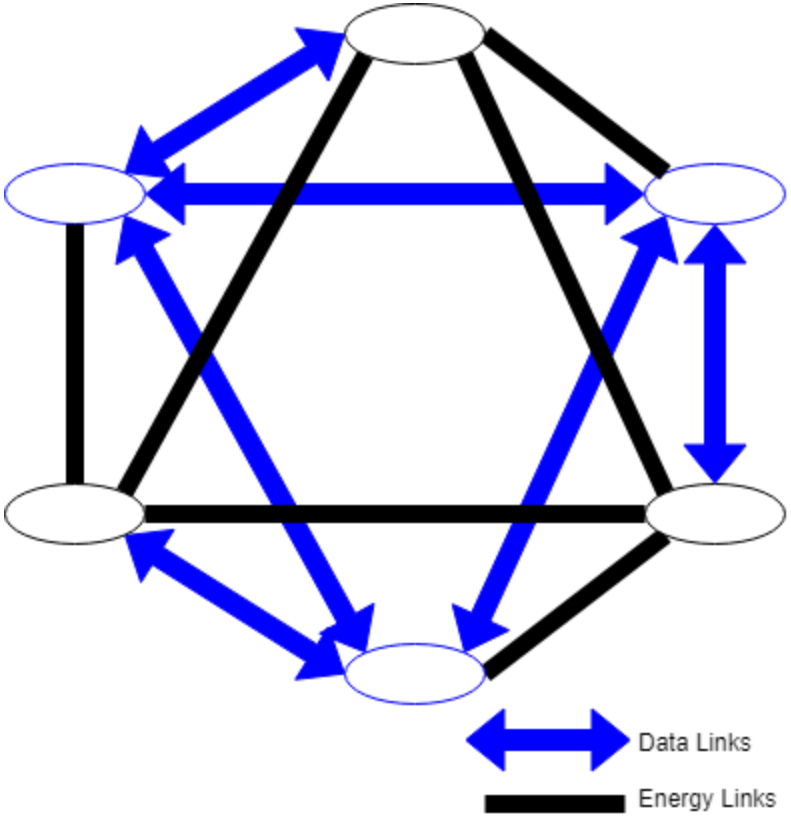
- RANalyzer remote spectrum analyzer (Xilinx Zynq Ultrascale+ ARM based)
- OTT/IPTV mesh Content Delivery Network gateway (AMD x86-64 based)
- blockchain-based cryptocurrency accepting vending machines
- “My Portable Digital Life” private cloud computing product
- multi-core / multi-ISA (e.g. ARM & RISC-V) run-time cross-checking
- run-time safety/security model compliance monitor for safe UAS in shared airspace
- **autonomous [cyber-physical] systems w/machine learning & mobile code**
- **DTN/IP/SDR gateway for B-LOS emergency networking**

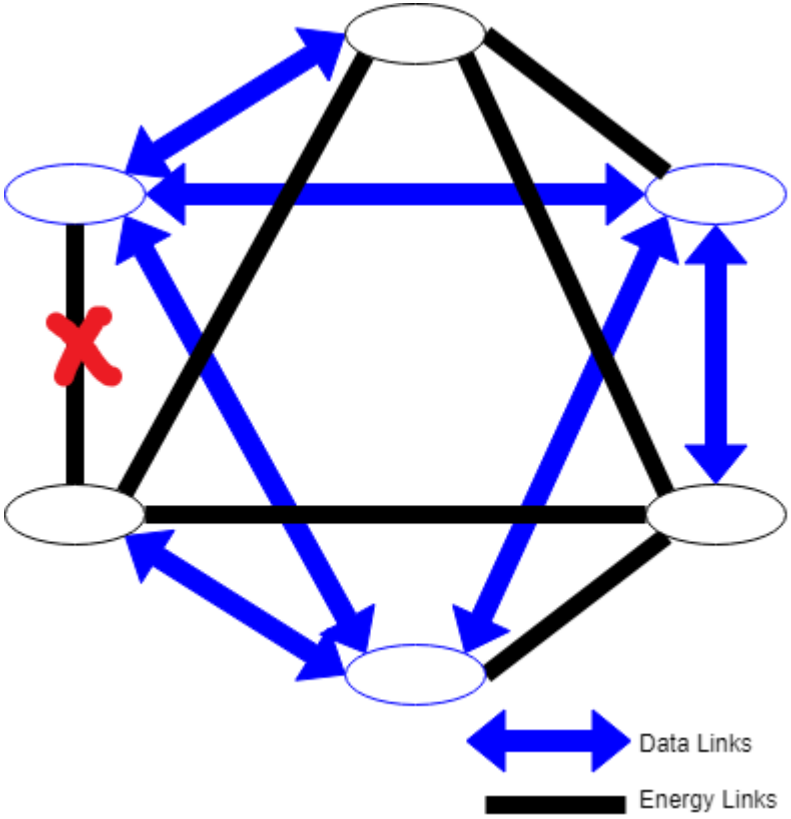
replicating autonomous agents



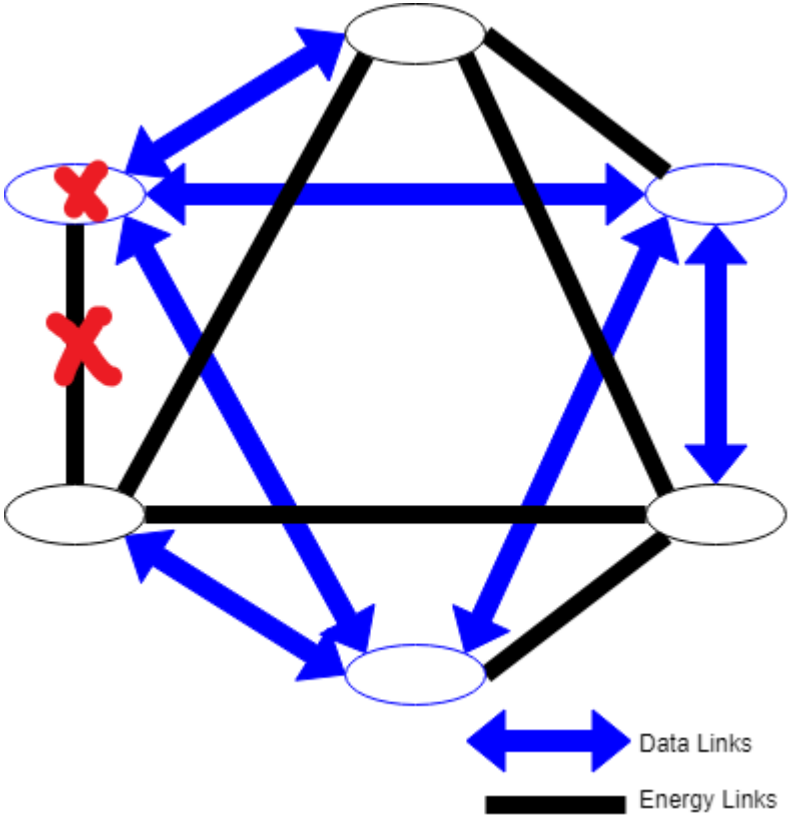
- mobile code can migrate nearer data sources
- useful / fit / "profitable" code can proliferate
- each resource is owned by 1 agent
- each agent demands credentials
- each agent follows its own policy
- some agents also learn [utilities]
- resource grants are capabilities

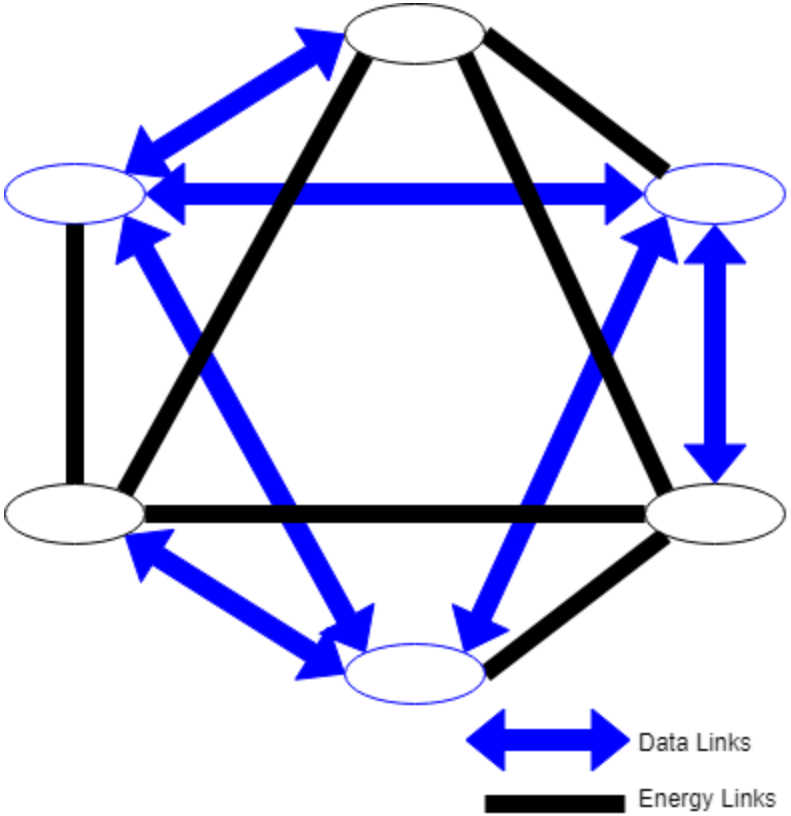
- credential/capability tickets required at each step
- not only data but also code can be transformed (by human programmers and/or Genetic Programming)
- *key issue is certification of transformed code:* credentials of parents do not apply to offspring, but may influence decision to trust offspring

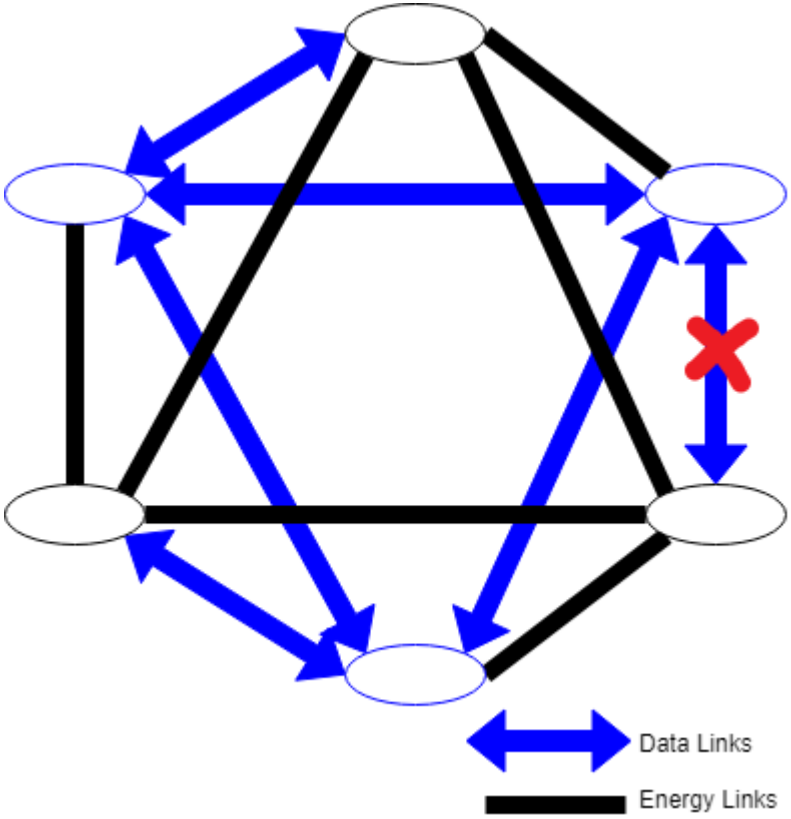


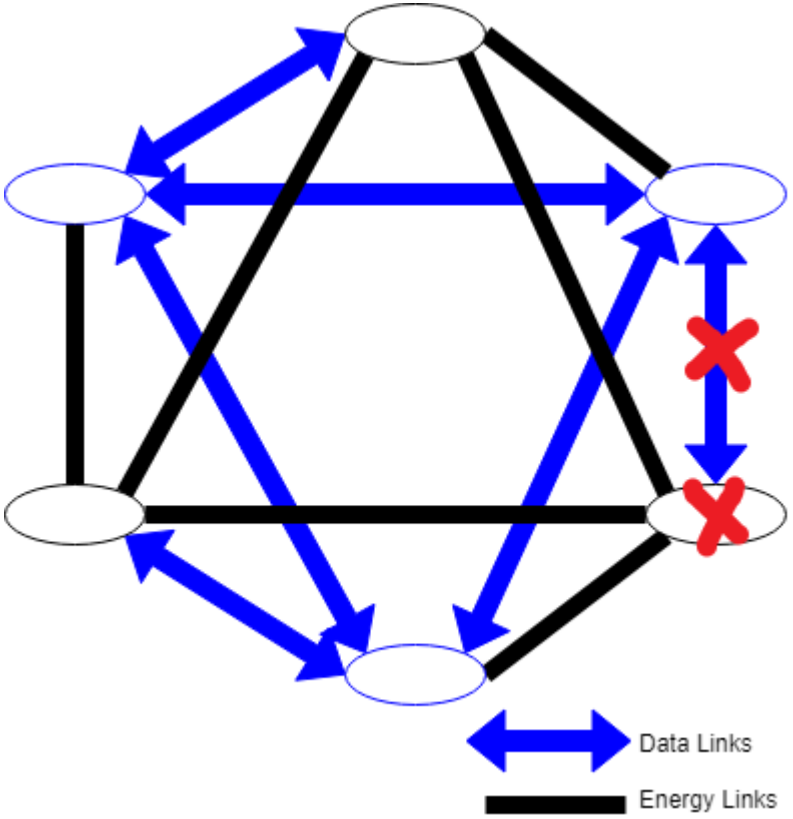


or knock out 2 core links; this energy distribution grid is very robust





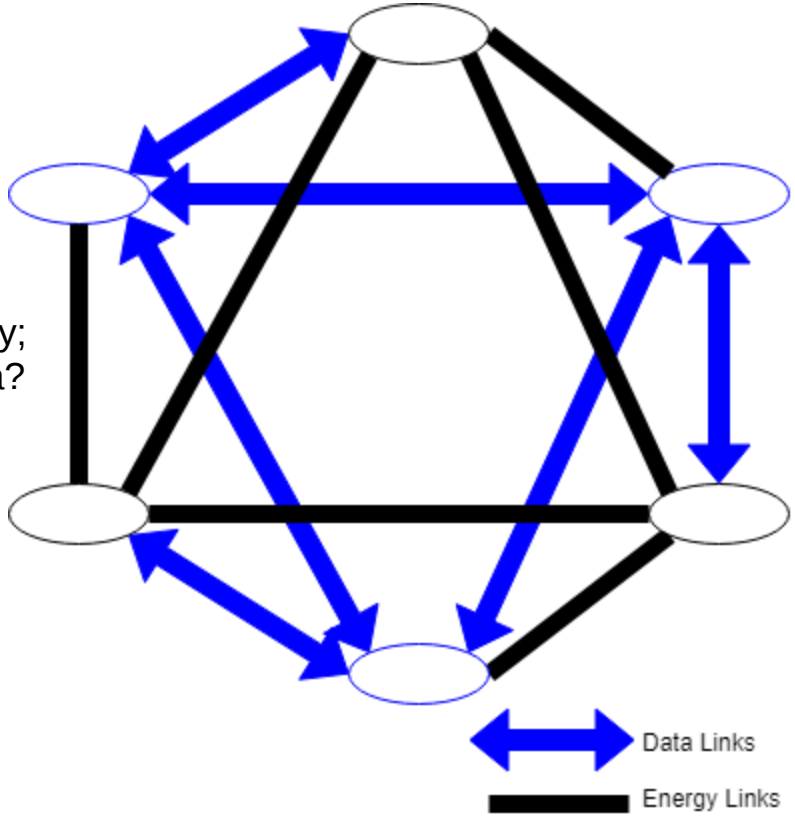


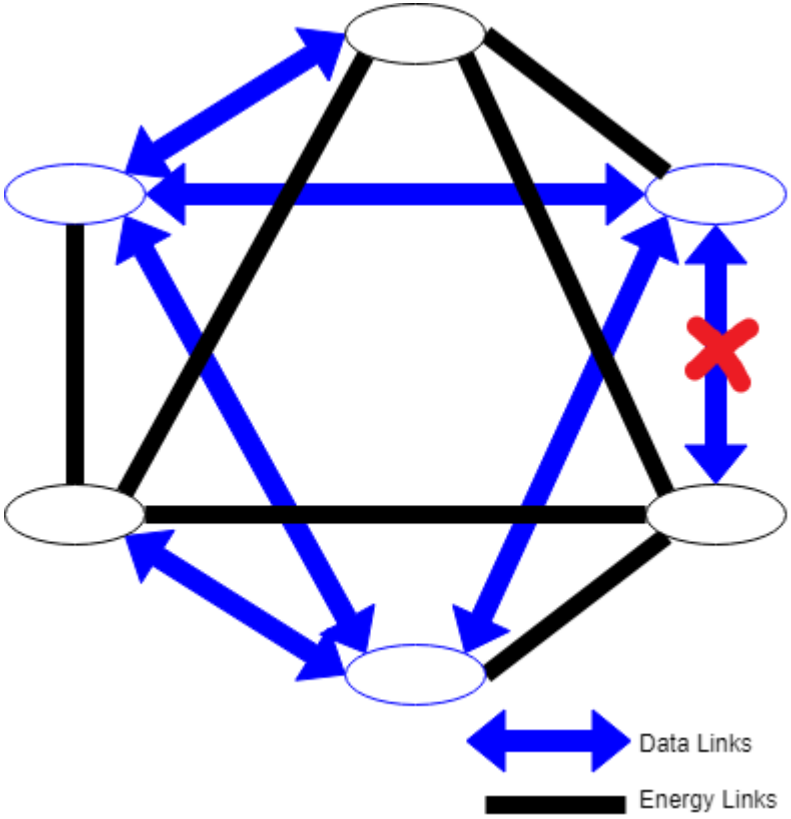


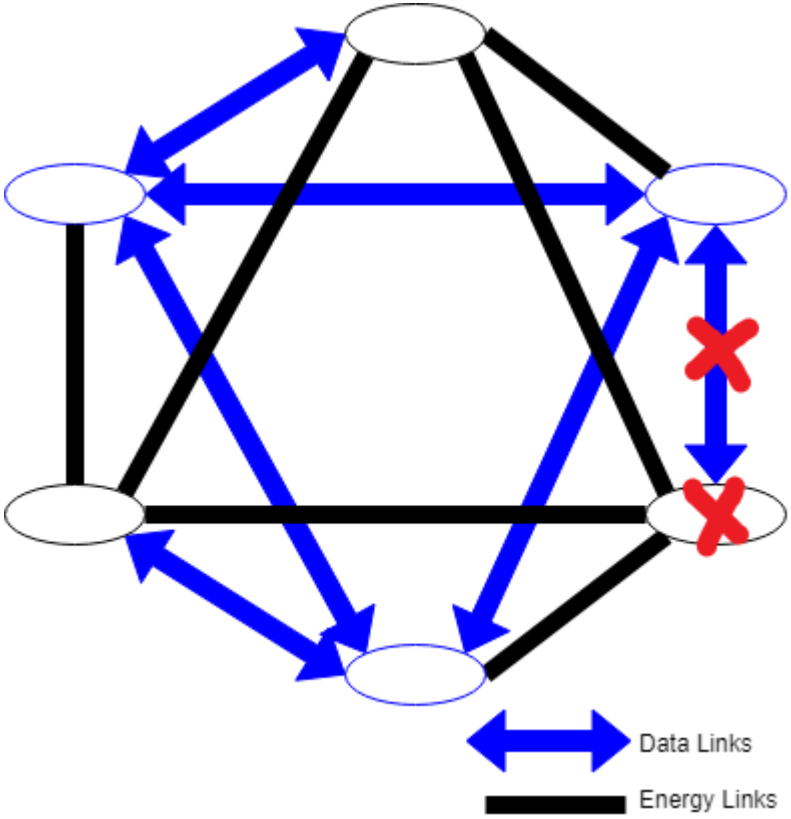
or knock out 2 core links; this data network also is very robust (it has the same structure as the energy distribution grid, rotated 60 degrees)

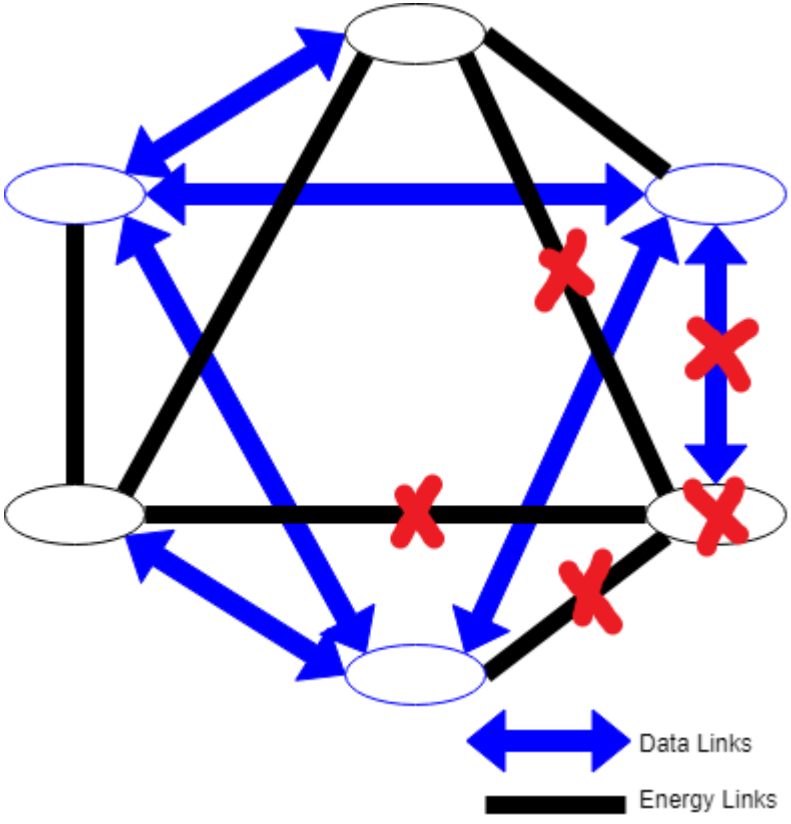
But what if:

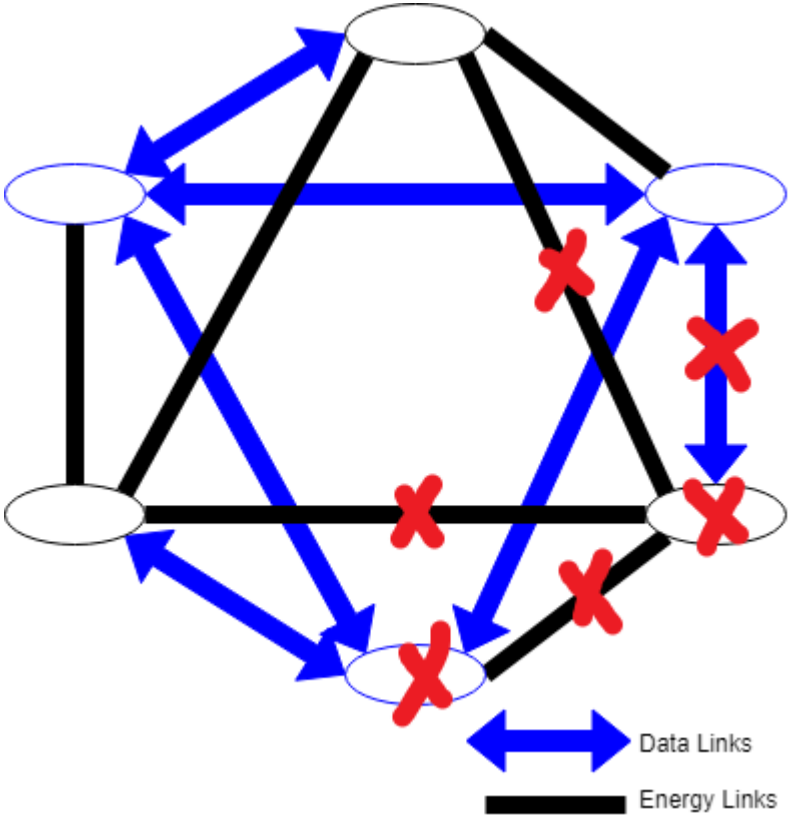
- data switches need energy;
- power switches need data?

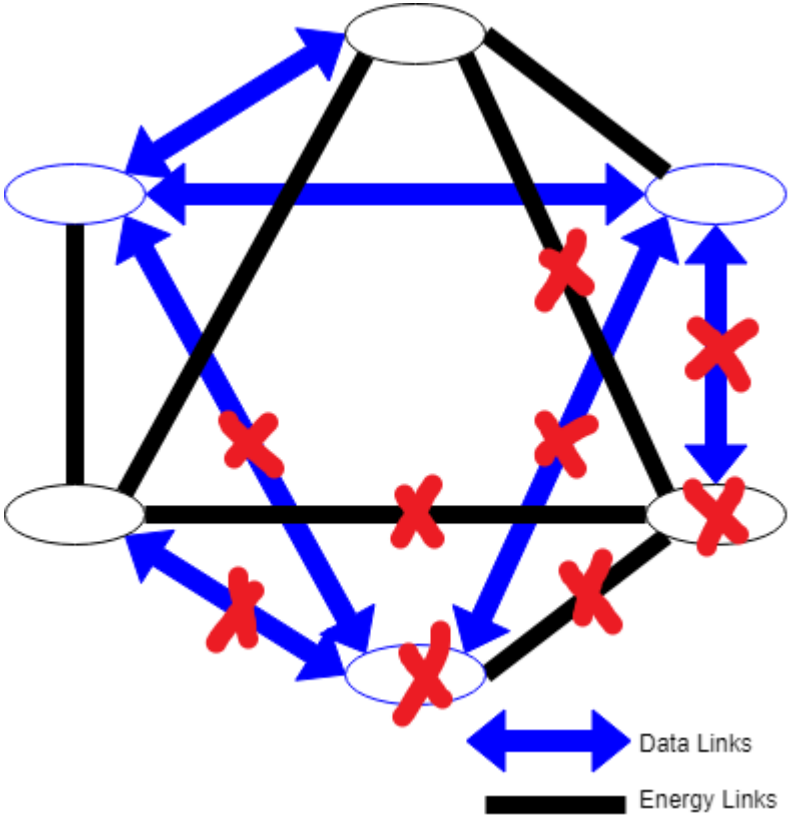


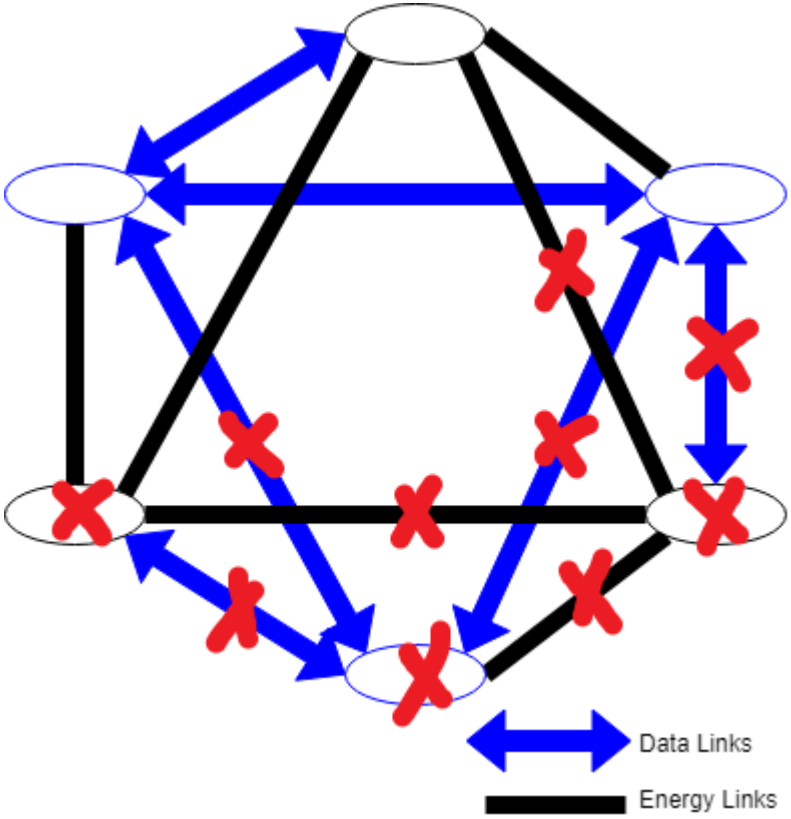


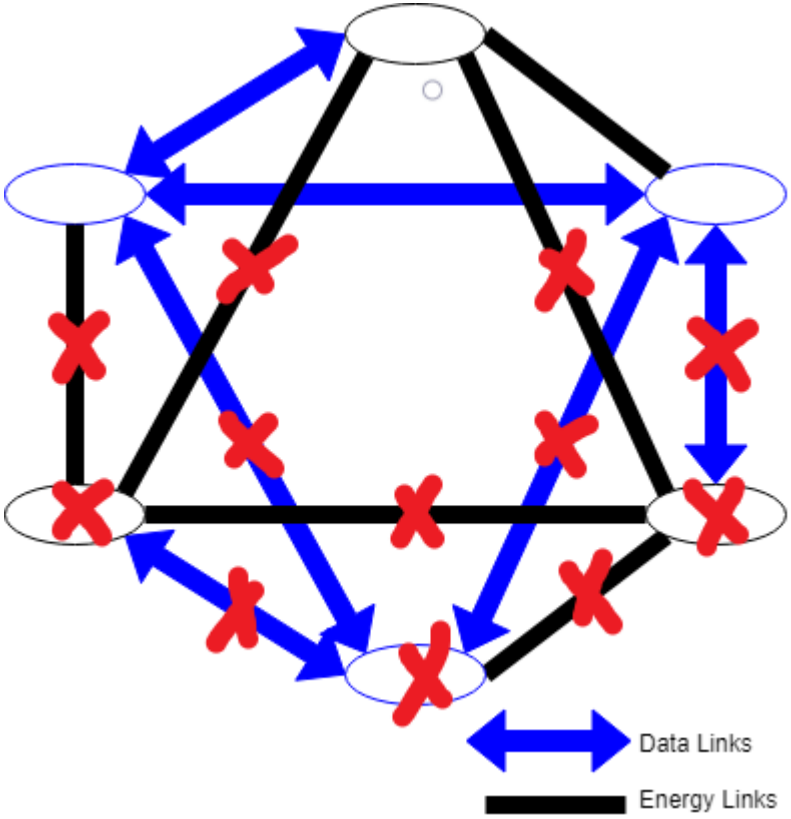


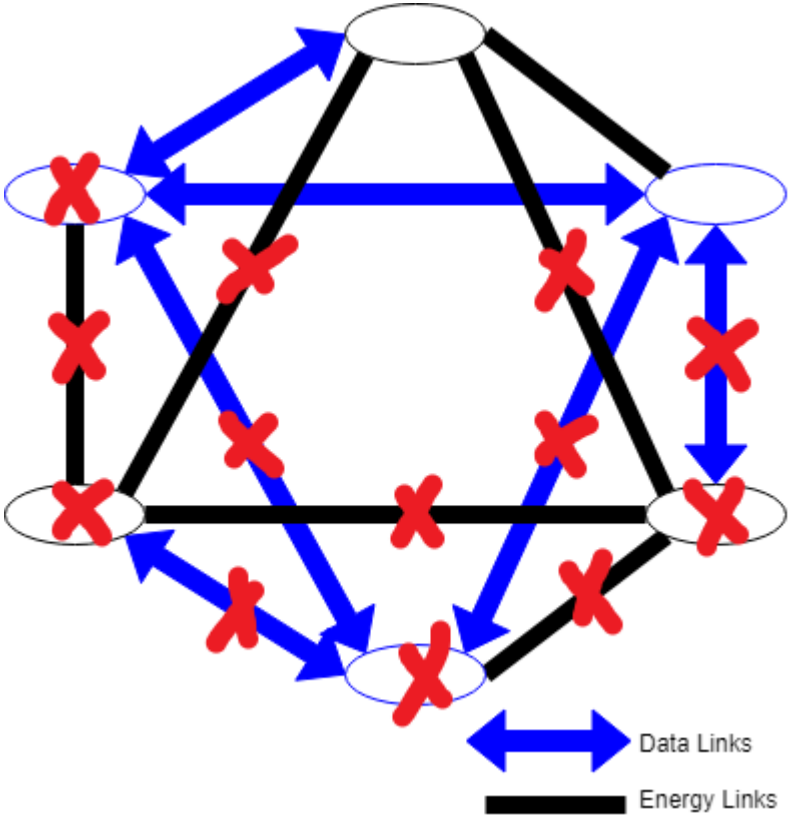


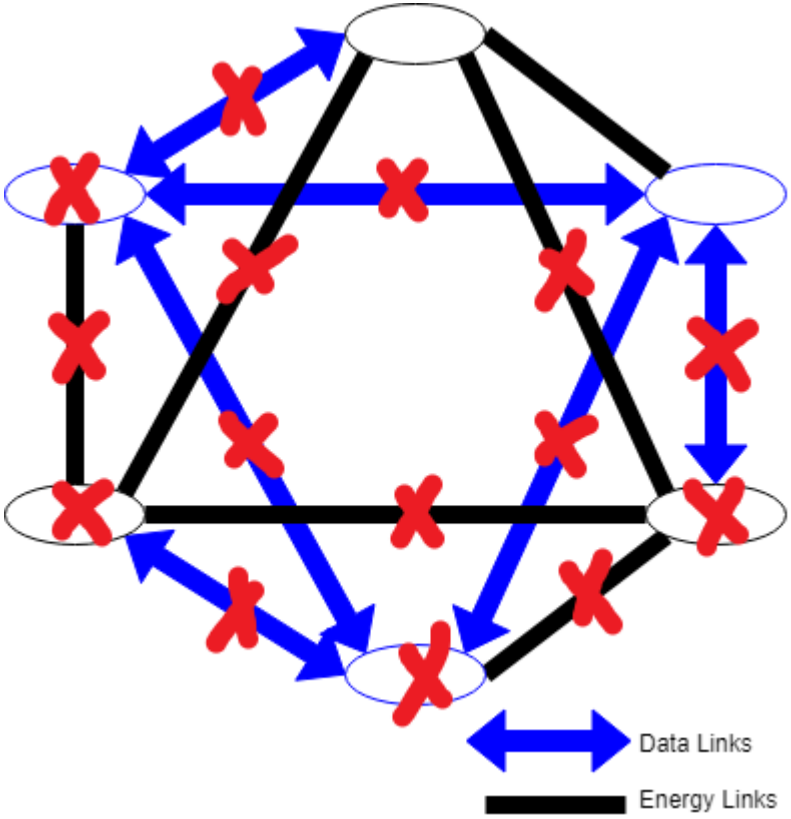


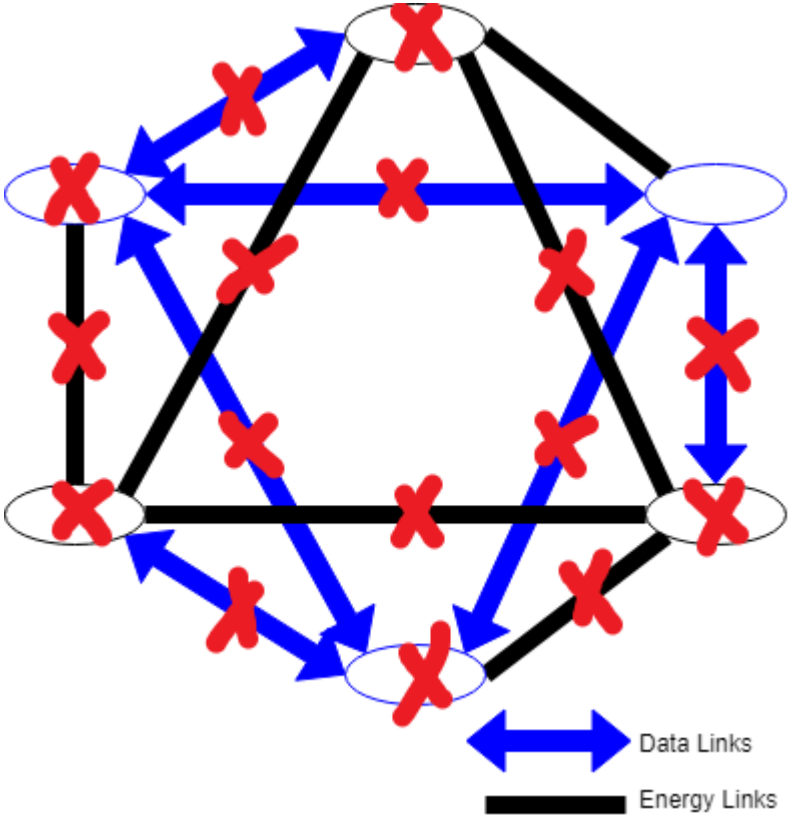


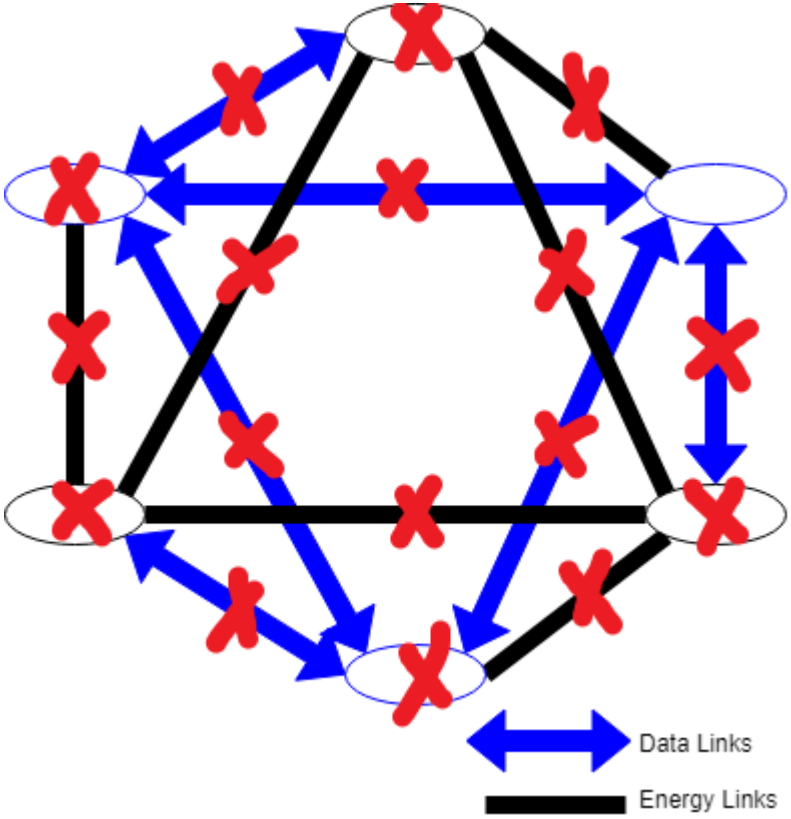


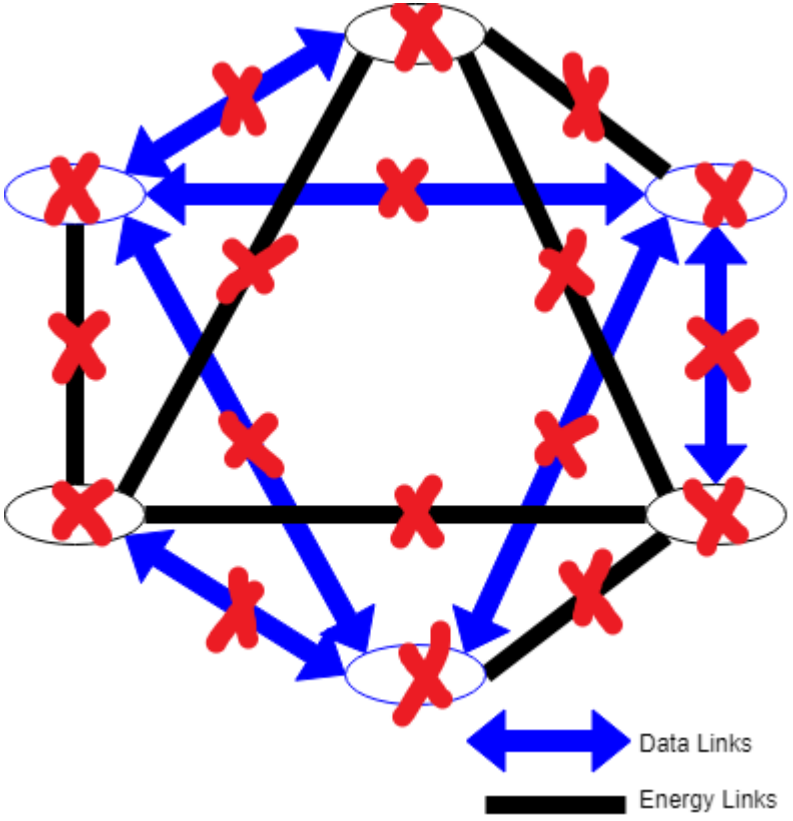












Cascading failures yield complete collapse of both separately robust networks, made fragile by introduced mutual dependence!

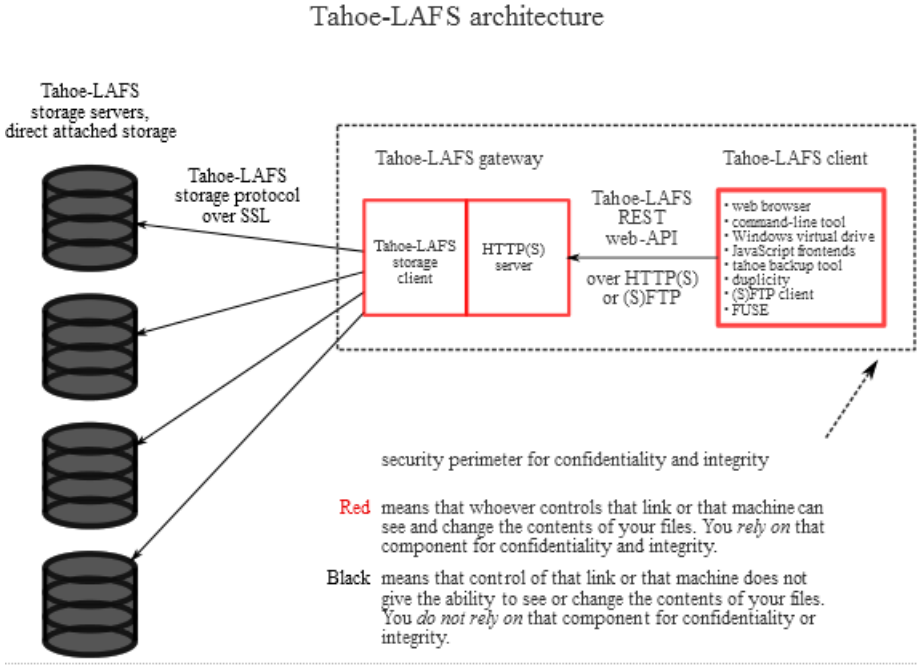
- many potential causes of widespread outage
  - cyber attack
  - solar Coronal Mass Ejection
  - nuclear ElectroMagnetic Pulse
  - https://www.whitehouse.gov/presidential-actions/executive-order-coordinating-national-resilience-electromagnetic-pulses prioritizes
- Our very small company is working unfunded efforts (we need not only funding but also technical help) to mitigate this existential risk by developing trustworthy Beyond Line Of Sight wireless gateways
  - WiFi, LTE CBRS, SATCOM, HF
  - local non-volatile storage
  - alternate power sources
- "guerilla" commercial deployment
  - IP Set Top Box (video etc.)
  - other nominal-condition services
- *many* issues to address
- seL4 helps w/some…

Solar & ham radio equipped Prius; photo used by permission of CDR (ret.) Bob Bruninga WB4APR

Wind & solar powered light pole, adaptable as Delay/Disruption Tolerant Network (DTN) / Internet Protocol (IP) wireless mesh router pole; photo used by permission of Colite Technologies www.colitetech.com

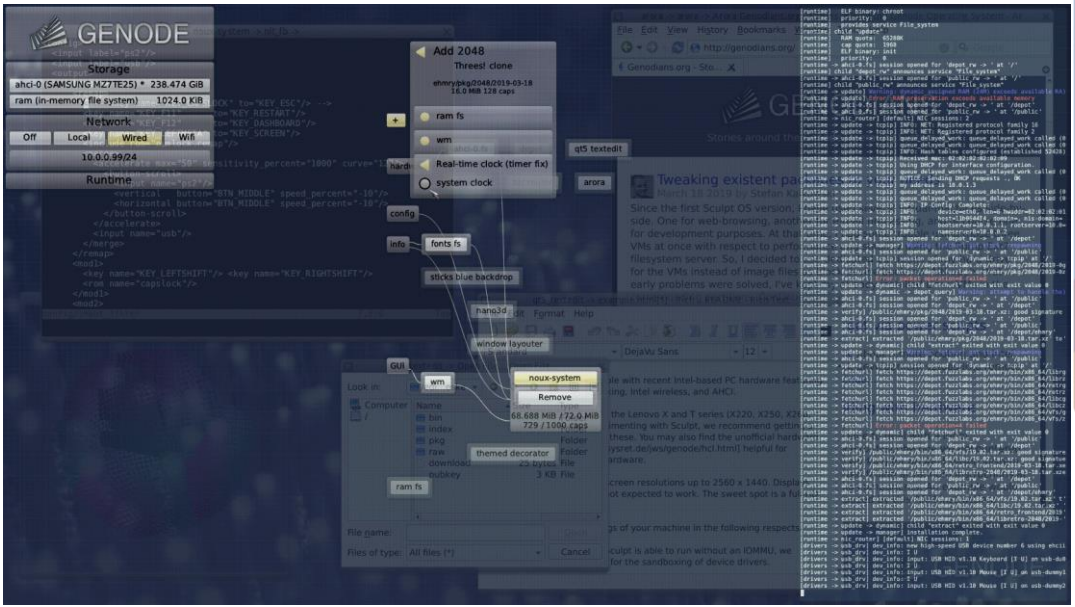
prototype gateway stack components (1 of 3)

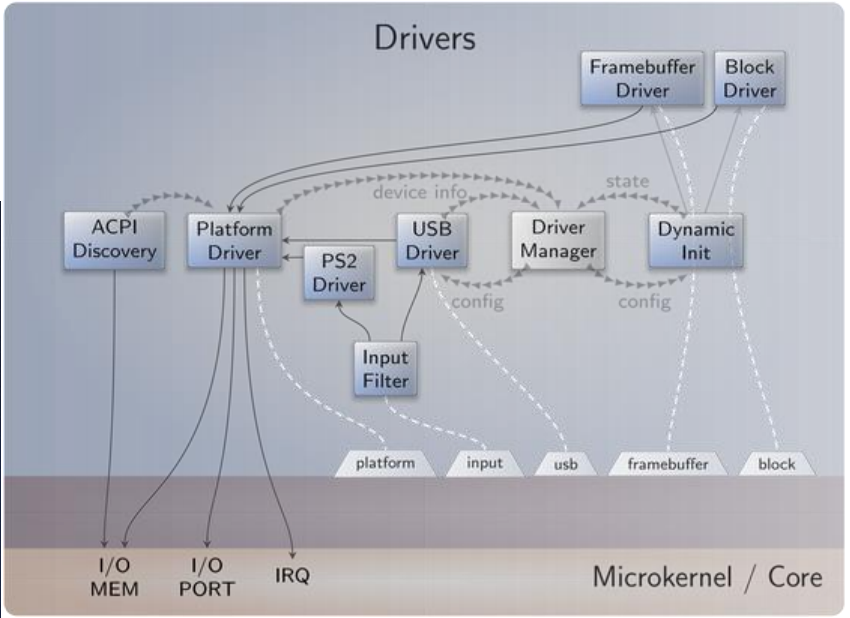
- Delay / Disruption Tolerant Network (DTN, RFC 4838 etc.) “Bundle Protocol” message router
- Tahoe – Least Authority File Store
  - ***k*** of ***n*** distributed storage
  - strongly encrypted
  - capability based access
- Host Identity Protocol (HIP, RFC 4423 etc.)
  - identifier – locator deconflation
  - HI private key is authentication capability
  - IP address binding is location capability
- GNU Radio Software Defined Radio (SDR)
  - XML specified graph -> auto-generated Python composition of C++ blocks
  - strong security needed for FCC approval
  - *natural for decomposing into components*



prototype gateway stack components (2 of 3)

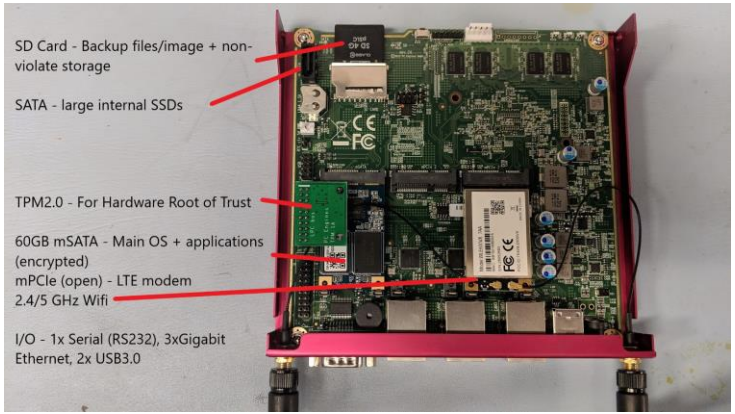
- component framework
  - VMs each on its own unprivileged VMM
  - or Sculpt natively component-oriented OS
- Genode (see slide after next)
- seL4

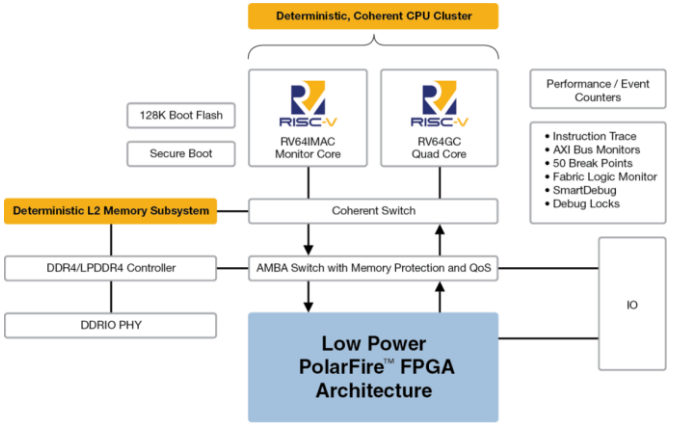




prototype gateway stack components (3 of 3)

- x86-64 w/Trusted Platform Module (TPM)
  - SABLE trusted/secure boot
  - our current primary prototype
- ARM
  - Xilinx Zynq Ultrascale+ MPSOC -> RFSOC
  - our SDR integration prototypes
  - we got Genode / seL4 running on real ARM hardware, inc. SabreLite (seL4 reference platform), rather than just in emulation
  - how-to etc. https://genodl4arm.critical.com/
  - code patches https://github.com/CriticalTechnologiesInc/genode-sel4-arm
- RISC-V
  - MicroSemi/MicroChip PolarFire FPGA SOC has other strong security features
  - in discussion w/several silicon vendors on adding Capability Hardware Enhanced RISC Instructions (CHERI)





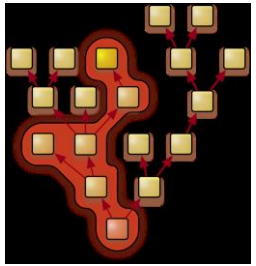
➢Least privilege: *capability-based security*
➢Mixed criticality
- *construct special-purpose OSes*
- *isolated components, app-specific TCBs*

➢Dependability
- *lightweight (~4 MB)*
- *account, trade, & track physical resources*

➢Scalability & Cross-Platform
- *recursive system structure*
- *host existing OSes, VMMs*
- x86, *ARM [TrustZone]*

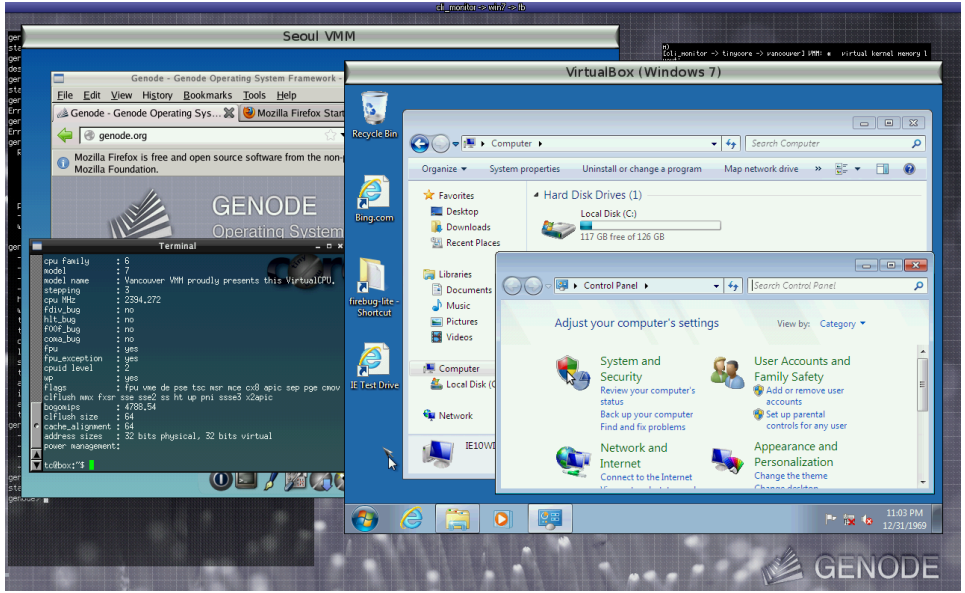
➢ Open Source Community Support
- *source code at GitHub*
- *several microkernels inc. seL4*
- *over 100 ready-to-use components*

Genode extends seL4 ecosystem & seL4 handles several issues for Genode, esp.
- *memory management*
- *inter-component communications*
- *formal foundation for capabilities*



Genode resources **dynamically** (vs CAmkES statically) allocated & accessed via capabilities strictly following process tree

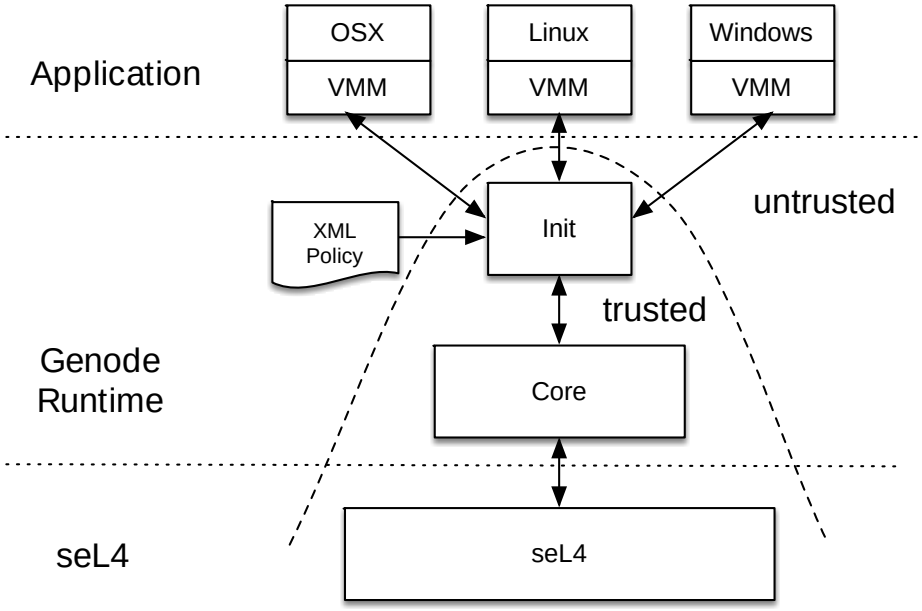
TinyCore Linux on Seoul VMM, MS Windows 7 on VirtualBox; both concurrently on Genode!



Genode + seL4: Mixed Trust Verification

DARPA supported work reported last year

- modeled Genode Core's behavior in the Alloy language/model checker
- showed that Core maintains authority confinement and integrity
- Genode caps backed by seL4 endpoint caps
- Syracuse University technical report https://sable.critical.com/media/extending-sel4-integrity.pdf
- formal models & proofs https://github.com/CriticalTechnologiesInc/GenodeVerified



But what about code that runs before seL4?

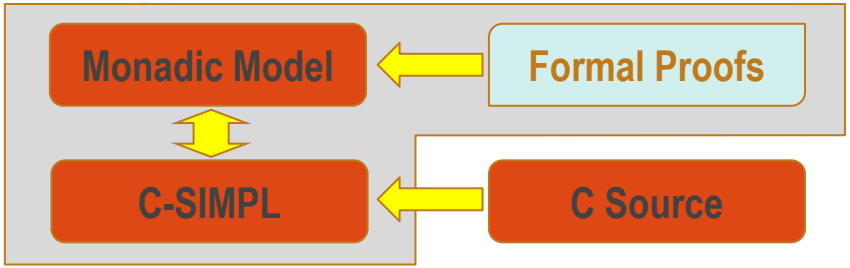
- “Yep, that’s one of the things #seL4 can’t protect you against: flaws in software that runs before #seL4 gets control. ” Gernot Heiser, 2018 NOV 06 https://twitter.com/GernotHeiser/status/1059995358317576192?s=19

- “These are attacks that can compromise the system even before the OS assumes control, eg. by *compromising the boot loader* or re-flashing the firmware, even seL4 cannot defend against attacks that happen before it is running.” (emphasis added) Gernot Heiser, paragraph 11 of https://microkerneldude.wordpress.com/2018/08/23/microkernels-really-do-improve-security/

- “… we assume that in-kernel assembly code is correct, hardware behaves correctly, in-kernel hardware management (TLB and caches) is correct, and *boot code is correct*.” (emphasis added) https://docs.sel4.systems/FrequentlyAskedQuestions#what-are-the-proof-assumptions

Syracuse Assured Boot Loader Executive (SABLE)

- S/W utilizes H/W "root of trust"
  - Trusted Platform Module (TPM chip) with "hardware protected capabilities"
    - 24 Platform Configuration Registers (PCRs) into which 160 bit hashes are "extended"
    - "seal" critical data (*e.g.* keys) to trusted state / unseal only in repeat of that state
    - SABLE uses both non- & -resettable PCRs to enable nested boot & Flicker
  - Dynamic Root of Trust for Measurement (DRTM) processor instructions
    - AMD skinit / Intel senter reset system to safe state & "late launch" Secure Loader Block
    - code in SLB is measured (hash extended into PCRs), measures & launches other code
    - *secure* boot precludes untrusted code gaining access to sensitive code, data, keys, etc.
    - *trusted* boot enables subsequent local & remote attestation of system configuration
- SABLE written to facilitate formal verification
  - Verified once by Syracuse University faculty & students
    - correspondence proofs between abstract and executable models of SABLE-unique code
    - nearly 4000 total lines of source code, about half in as-yet unverified crypto libraries
    - labor intensive; approximately 20-30 Lines of Proof for each Line of Code
  - Released as open source in hopes of independent verification & experimentation
    - tech reports https://surface.syr.edu/eecs_techreports/182/ & https://surface.syr.edu/eecs_techreports/183/
    - source https://github.com/CriticalTechnologiesInc/qemu-tpm-svm & https://github.com/CriticalTechnologiesInc/SABLE
    - proofs https://github.com/CriticalTechnologiesInc/SableModel
    - configuration instructions, developers notes, etc. https://sable.critical.com/



seL4 microkernel itself does not use a conventional heap but some code running atop seL4 might need one; tools used by Data61 to verify seL4 need a heap allocator satisfying certain properties for those tools to be used to verify code using a heap; we wanted a heap in SABLE also, so wrote one (part of SABLE open source release), you might find it useful.
We verified our heap allocator using Isabelle/HOL.

Procedure

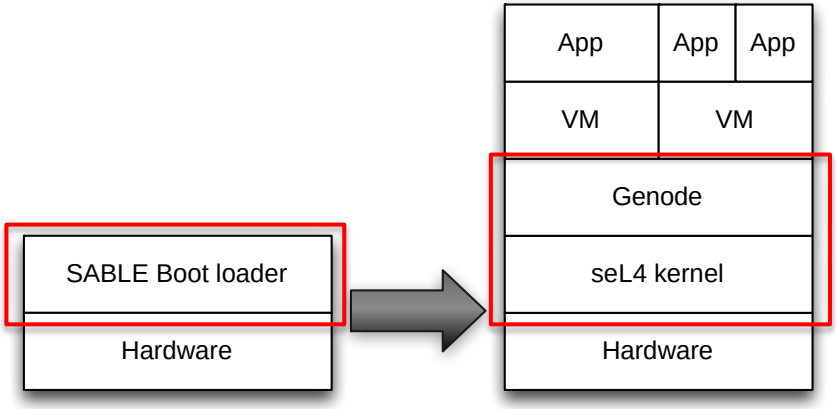
1. We parse the C source code into a primitive imperative language called C-SIMPL, which precisely models C memory semantics.
2. We lift the C-SIMPL code into imperative monadic code using the NICTA AutoCorres tool, which automatically proves that the C-SIMPL code exhibits the same behavior as the abstracted code.
3. We use a Verification Condition Generator to prove Hoare triples over the alloc function.
4. Bugs in the code will prevent us from proving Hoare triples. When this happens, we fix the bug and go back to Step 1.

Verified Properties

- Calls to alloc cannot fail (guaranteed to return).
- Allocator correctly calculates available memory.
- All “free” areas really are.
- alloc() cannot corrupt the heap.

Results: discovered & fixed 2 previously unrecognized vulnerabilities/bugs (integer overflow & improper comparison)

Secure Boot



- Boot if and only if
  - We can mutually authenticate System and User
  - We are booting a previously “blessed” configuration
- **Only capability-based secure boot of which we are aware**
  - if TCB measurements wrong, not just “will not complete boot”, but also “cannot complete boot”
  - essential crypto keys derived from: PCR values; operator input; TPM hardware-protected capability operation
  - Full Disk Encryption (FDE) of not only ‘/’ (root) but also ‘/boot’ (kernel, initramfs)

how hardware root of trust enables secure & trusted boot

- Trusted Platform Module (TPM) 2.0: tamper-resistant crypto coprocessor w/secure storage
  - "measure": stored strong crypto hash of a component makes undetectable changes to it infeasible
  - "extend": chaining together sequences of hash values into PCRs protects entire TCB configuration
- Dynamic Root of Trust for Measurement (DRTM) "late launch" CPU instruction (e.g. AMD skinit)
  - resets [running] system & puts 0 in dynamic PCRs

  (only instruction other than hard reset that can force a specific value into a dynamic PCR)
  - measures Secure Loader (SL), extends measurement into specified PCR
  - unconditionally launches SL
- seal(): encrypt data using key & state of designated PCRs
- unseal(): decrypt data using key & state of designated PCRs

$$\text{PCR} \leftarrow H(0||H(SABLE))$$

$$\text{PCR} \leftarrow H(PCR||H(seL4))$$

$$\text{PCR} \leftarrow H(PCR||H(Genode))$$

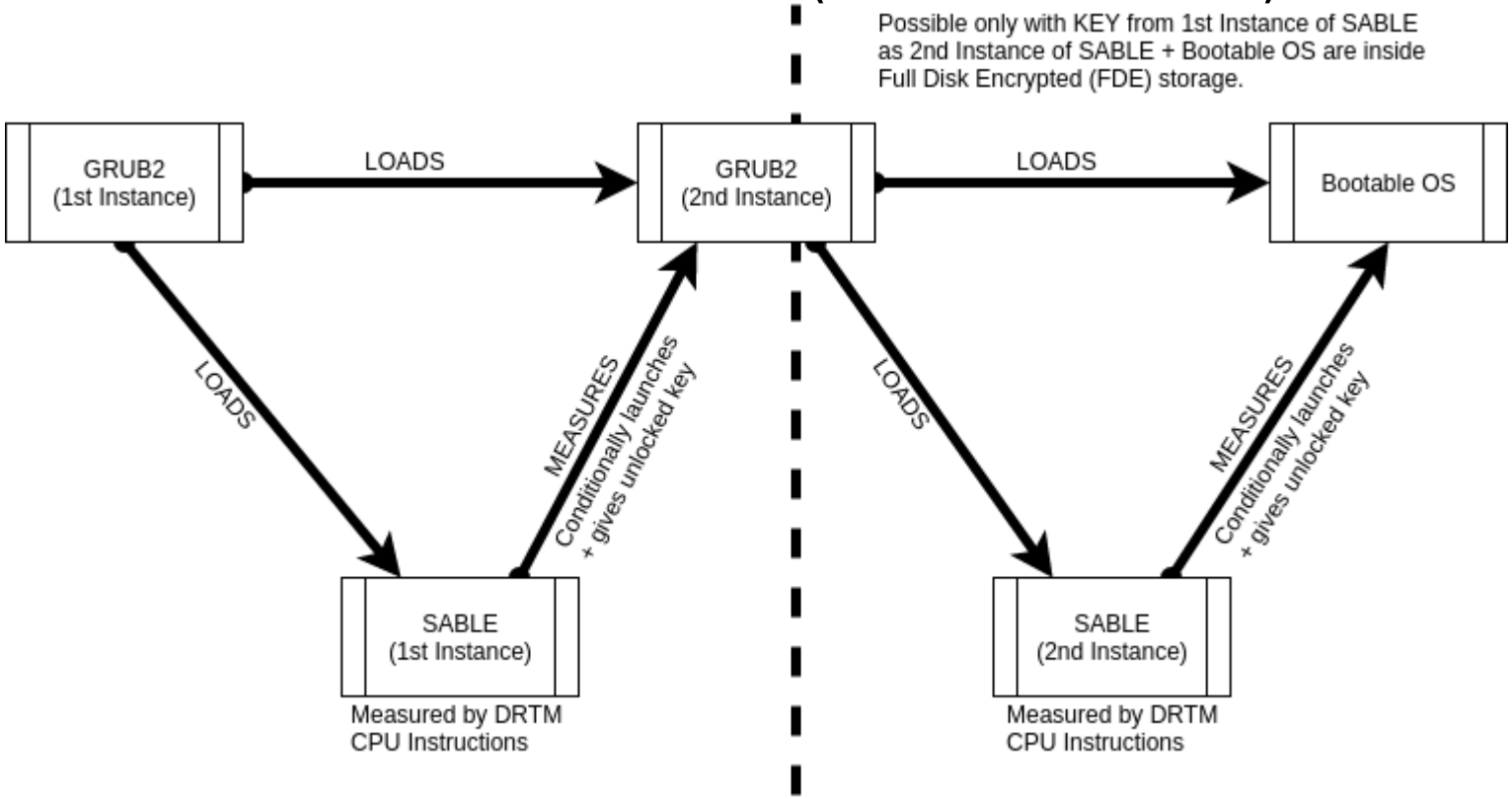
SABLE Secure Boot

- Capability-based
  - Not just “will not complete boot”
  - But also “**cannot** complete boot”
- Essential crypto keys derived by TPM hardware operation combining
  - PCR values
  - Operator input
  - TPM private key that cannot escape the TPM hardware
- Full Disk Encryption (FDE)
  - Encrypts not only ‘/’ (root)
  - Encrypts also ‘/boot’ (kernel, initramfs)
  - P. Kogan, “Full disk encryption with LUKS”, http://www.pavelkogan.com/2014/05/23/luks-full-disk-encryption/
- Updated Grub to prompt user to decrypt LUKS volume on boot

➢ **Only capability-based secure boot of which we are aware**

➢ **Enables users to know the TCB running Monday is the previously approved TCB that was running last Friday**

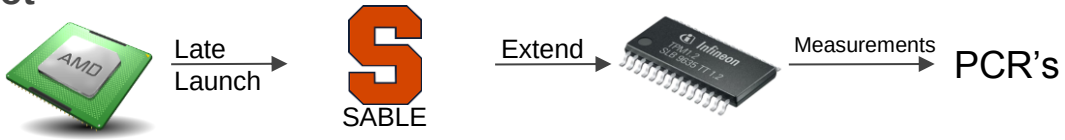
Overall SABLE Boot Process (can be further nested)



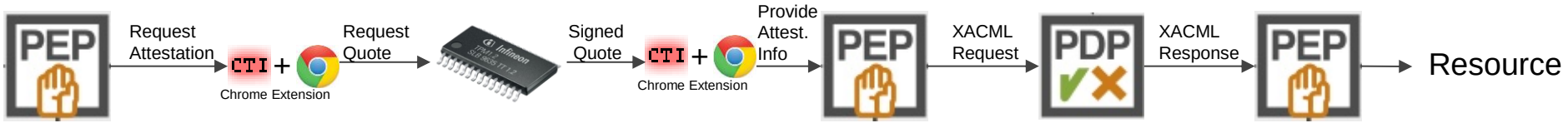
XACML/SAML based “AAA”

(Attestation, Authentication, Authorization, Access Control, Attribution, Accounting & Audit, initially for a distributed repository supporting a publish-subscribe messaging system)

On Boot



On Attestation

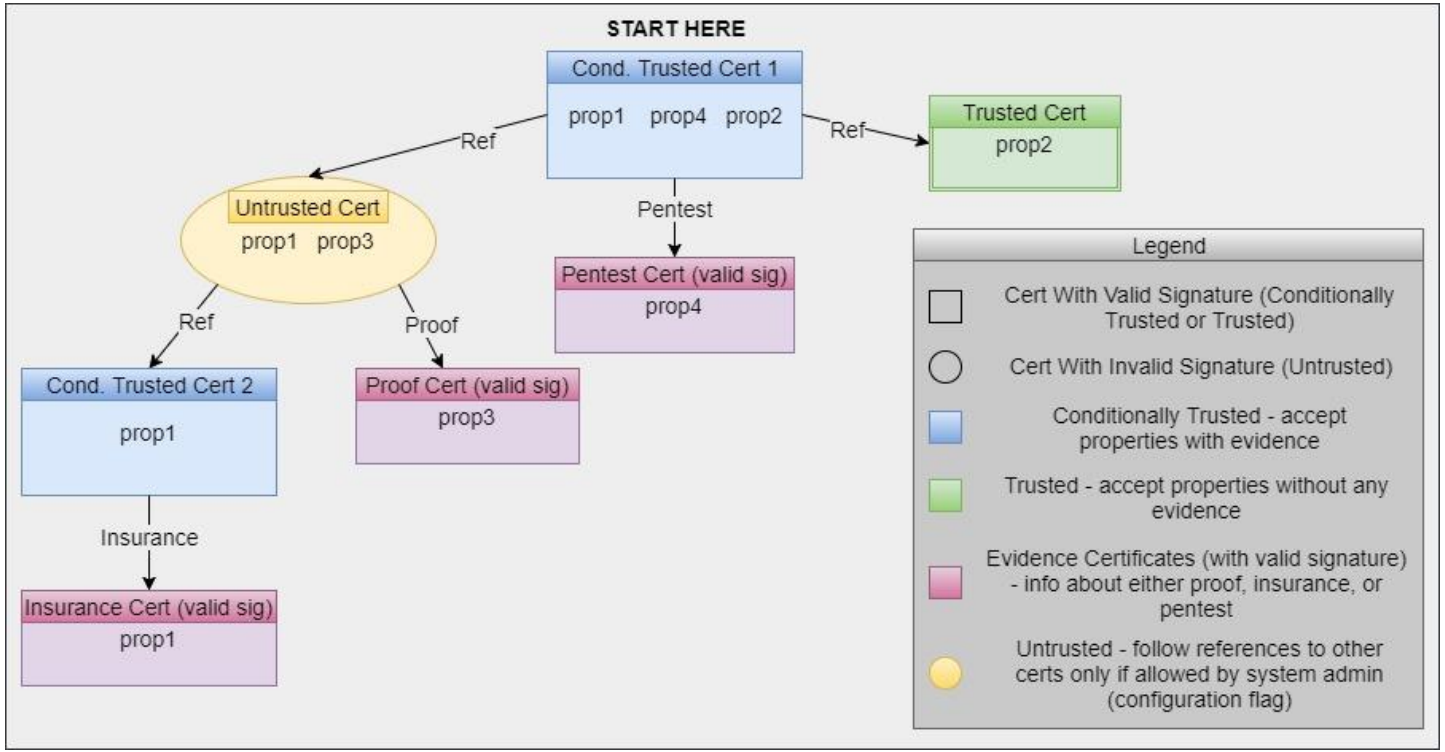


Attestation

attestation of trustworthiness of Damon & Pythias in Greek myth
painting by Eloi-Firmin Féron
lo-res scan of unknown origin,
uploaded 2013, Public Domain,
https://commons.wikimedia.org/w/index.php?curid=30306878

- "TPM quote" from a Trusted Platform Module
- Arbitrary set of Platform Configuration Register (PCR) values as they currently exist in the TPM, subsequent to them being "extended" sequentially with measurements of TCB elements, plus a nonce, signed by a pseudonymous Attestation Identity Key (AIK)
- XACML driven attestation process
  - Quote is validated & verified by server Policy Enforcement Point (PEP)
  - After validation & verification, list of binaries comprising the quote are known, which are then used later on to look up known properties about them. If no properties are known, user is prompted to provide a URL to a certificate that asserts properties.
  - PEP then forms a request as usual, but now including a list of binary hashes, and sends to Policy Decision Point (PDP)
  - PDP, by means of a PIP (Policy Information Point) looks up which security properties are associated with each binary, which in turn is then used to see if a user satisfies that portion of policy

Walk the security property evidence certificate tree *at access time* to determine whether policy is satisfied.



“drivers”: device driver software

All know that the paucity of hardware driver software for seL4 is still limiting; our most pressing needs are –

- TPM 2.0
  - need a Virtual TPM for each VM
    - PCR extend operation is sequential
    - does not support branching for multiple concurrent VM-based TCBs
  - beneath that, need a real TPM 2.0 hardware driver for Genode/seL4 framework
  - started by getting Linux tpm2tools running on our initial prototype x86 hardware
  - decided tackling TPM as our 1st device driver was too hard
- network interfaces
  - need hip0 tunnel driver to enable strongly encrypted, mutually authenticated, inter-VM communications
    - Indeed, can hide raw eth0 from VM, force it through hip0 so it can only interact within HIP overlay
  - beneath that, need virtual eth0 driver for VMs
  - beneath that, need real eth0 driver for Genode/seL4
  - decided all these were also too hard for us n00bs ;-)
- PCIe serial card
  - working on it! Hard slogging but learning…

“drivers”: lessons learned

- The learning curves of Genode and seL4 are substantial: it took nearly one year for several junior programmers to get the framework and microkernel running together on x86-64 and separately on ARM.
- Although progress is being made on 64 bit ARM, virtualization on x86-64 and IOMMU with each, it is slow.
- Although progress is slow, code change is rapid, causing bit rot of formerly working configurations.
- What works on an emulator may not work on real hardware (not newly learned, but confirmed).
- What works on a hardware based TPM may not work on a firmware based TPM (likewise).
- Run-time dynamic loading/linking in a microkernel based environment is hard; Genode/CAmkES hybrid?
- She swallowed a spider to catch the fly: seL4 needs a secure bootloader; SABLE needs admin utilities; probably we need to formally verify the admin utilities as they configure bootable item security? ☹

Current Transition / Future R&D in Mixed-Trust Systems

- commercial product development clients
  - RANalyzer remote spectrum analyzer (Xilinx Zynq Ultrascale+ ARM based)
  - OTT/IPTV mesh Content Delivery Network gateway (initially AMD x86-64 based)
- IR&D
  - DTN/IP/SDR gateway [for B-LOS emergency networking] described herein
  - blockchain-based cryptocurrency-accepting vending machines (patent expected to issue soon)
  - My Portable Digital Life (see graphic below) [w/cryptocurrency wallet]
- seeking sponsors
  - multi-core / multi-ISA (e.g. ARM & RISC-V) run-time cross-checking
  - run-time safety/security model compliance monitor for safe UAS in shared airspace
  - autonomous cyber-physical systems w/machine learning [& mobile code]
- https://opensrc.critical.com/ has all our open source code, model, proof, docs, etc. releases
- contact us www.critical.com 315-793-0248 stu.card@critical.com

damaged? lost? stolen? tampered? no problem!




rapidly restore hardware, data & services