



Systems and Internet Infrastructure Security

Network and Security Research Center
Department of Computer Science and Engineering
Pennsylvania State University, University Park PA

A Case for Building Integrity- Verified Channels

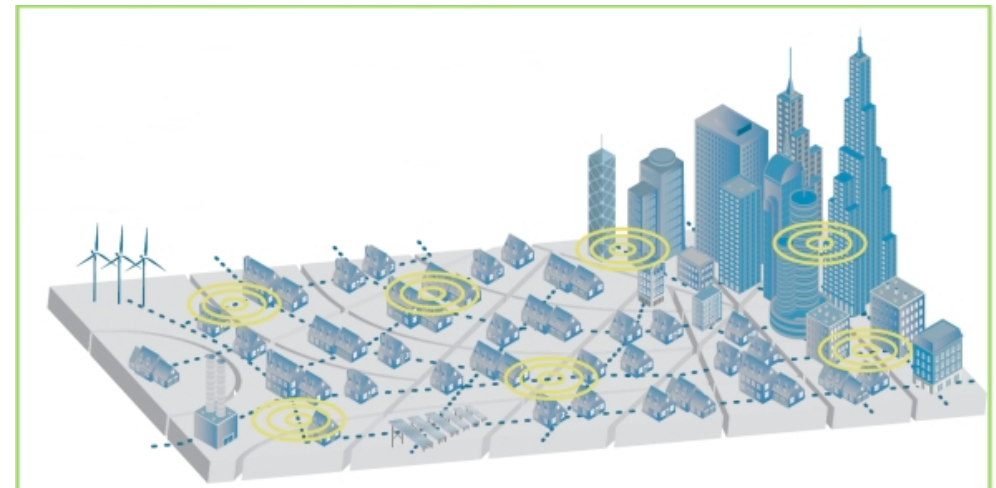
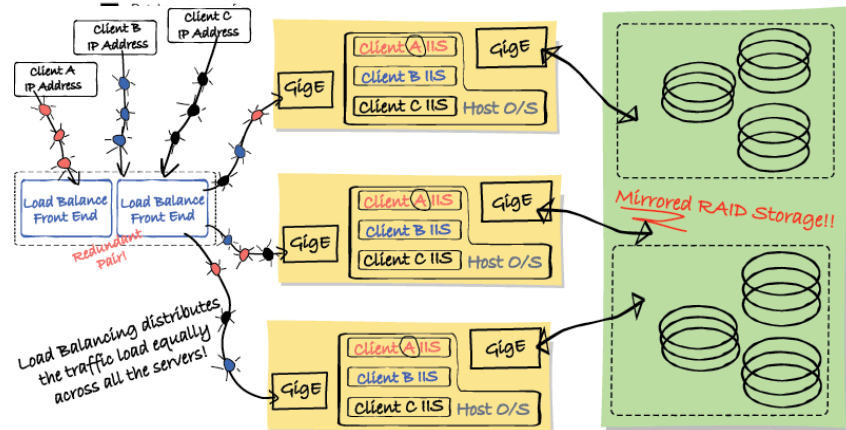
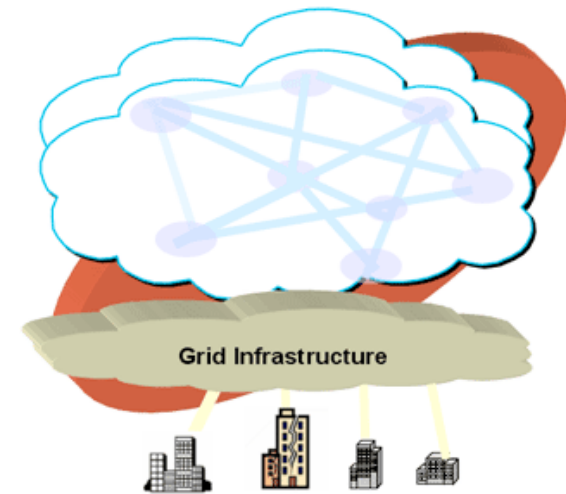
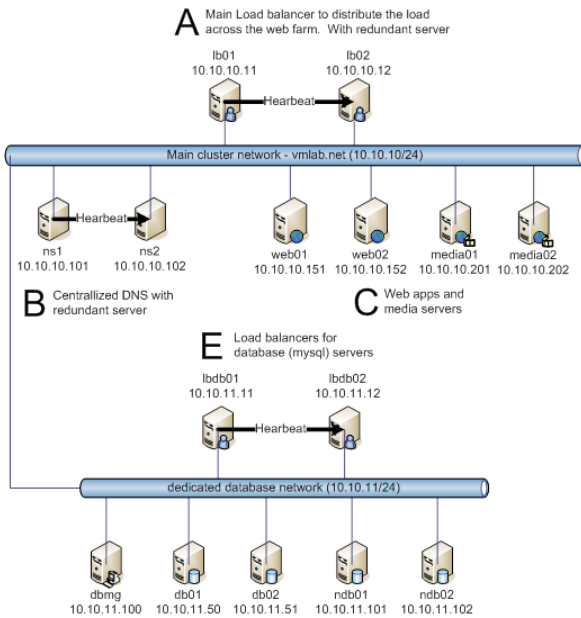
Trent Jaeger

*Systems and Internet Infrastructure Security (SIIS) Lab
Pennsylvania State University*

May 5, 2009

Distributed Systems Revolution

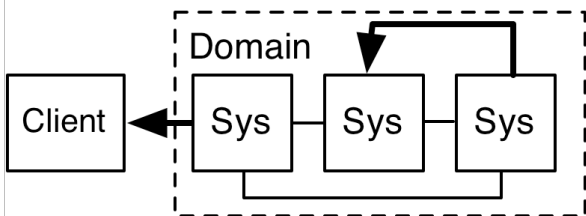
- Computing is going global



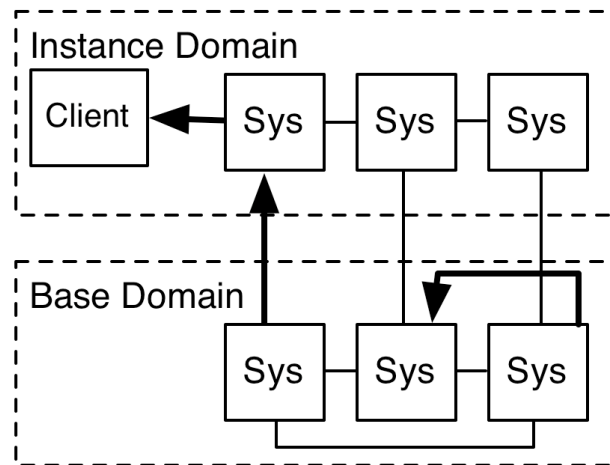
- *How do I know that the remote computation has been executed correctly?*
 - ▶ In client-server web applications, we trust the server
- *Lots of instances of server misconfiguration in past*
 - ▶ Banking website security (e.g., use of https) is often incorrect
- *Now, the correct operation of my business depends on me and someone else working together correctly*
 - ▶ Not just ignorant users who will blindly trust servers
- ***Systems Must Be Capable of Justifying Their Integrity!***

Distributed System Architectures

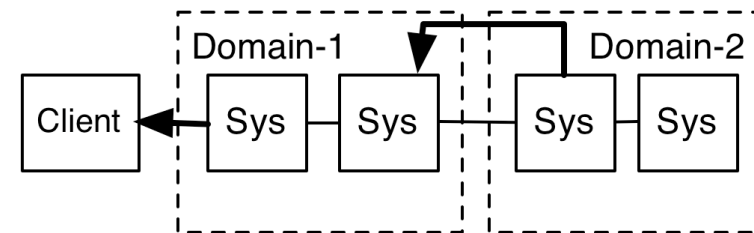
- We view distributed computing systems in the following ways



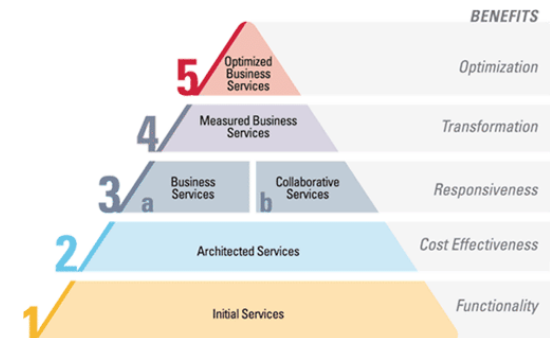
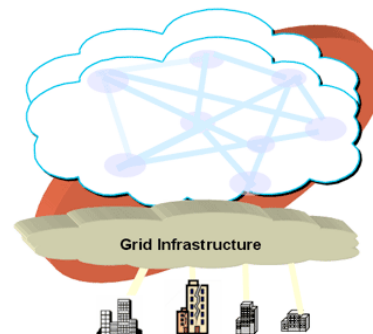
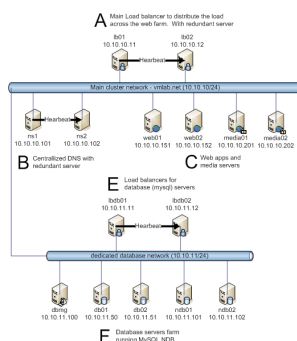
Single Domain



Horizontal Domains



Vertical Domains



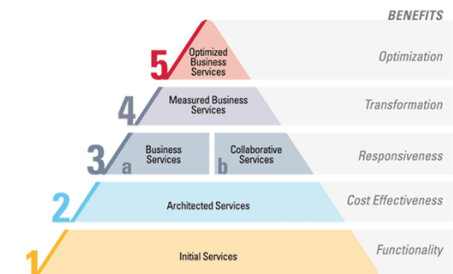
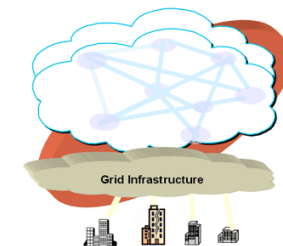
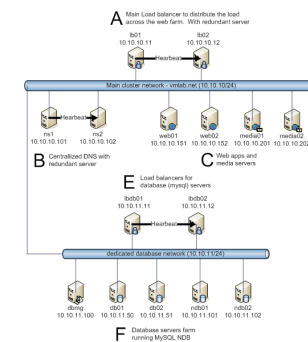
Why Should We Care?

- A scenario – **Remote Medical Procedure**
 - ▶ *Would I trust my life with these things?*
- Normal case
 - ▶ Doctors are local, isolated from attackers, correctly “configured”
- Remote case
 - ▶ Servers provide instructions
 - ▶ Locally performed



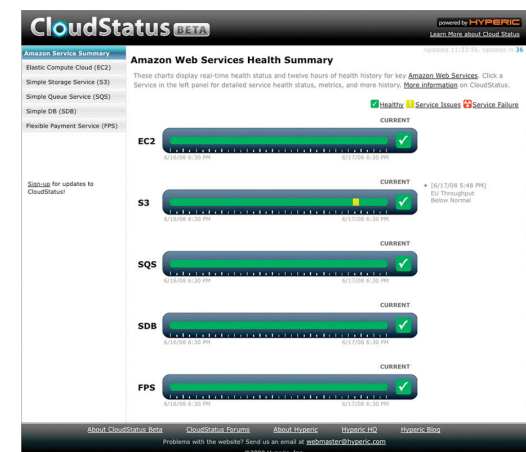
Security Requirements

- Server is configured to run medical application that determines instructions
- Before we would let this be done...
 - ▶ Physical security
 - ▶ Secure communication
 - ▶ Root of trust (e.g., TPM)
 - ▶ Data security
 - High integrity computation
 - Protect data from leakage



Cloud Security

- For Amazon's EC2
 - ▶ *Physical security* – “military grade perimeter”
 - ▶ *Request* – Clients sign all requests
 - ▶ *Certification* – Compliance with SOX and HIPAA are aims
 - ▶ *Bastion Host System* – Client-authenticated communication only
 - ▶ *Cloud Host System* – with Bastion only, customized Xen
 - ▶ *Guest OS* – Up to customer
 - ▶ *Firewall and Storage* – Communicate via controlled network and storage
- “... better than many IT systems”



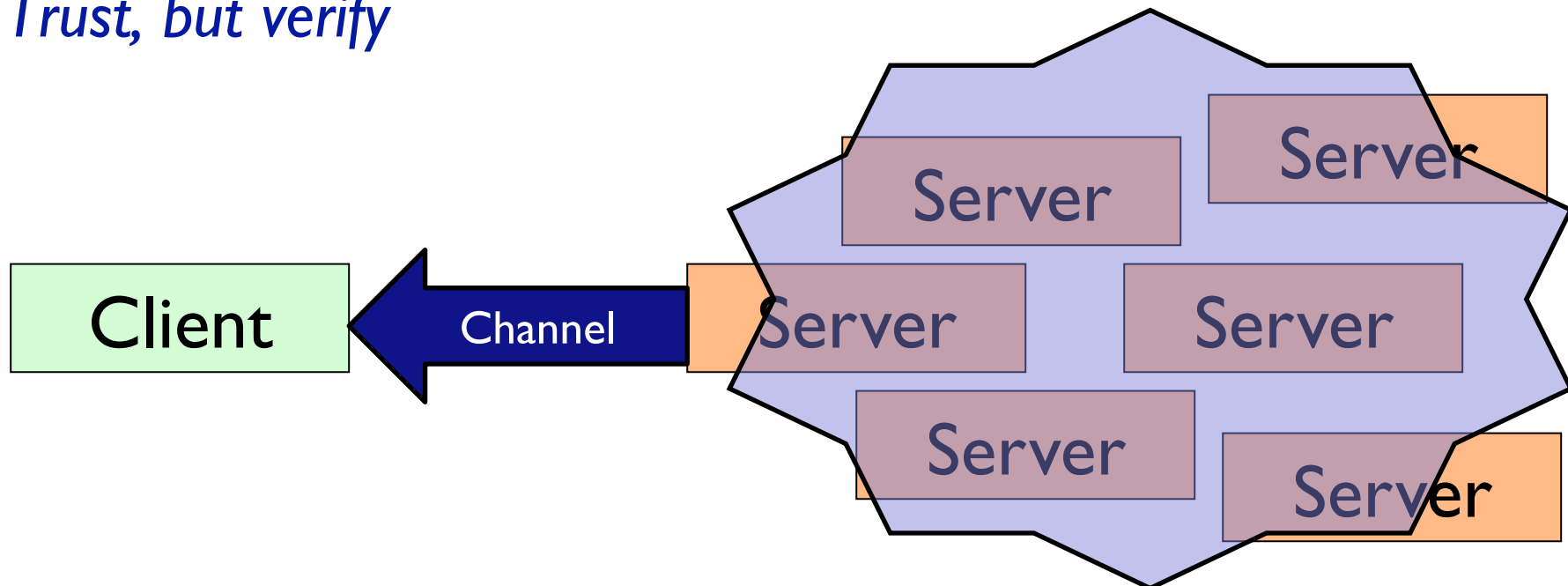
Cloud Security Challenges

- For Amazon's EC2
 - ▶ *Request* – A protocol bug took 7.5 months to fix
 - ▶ *Host Systems* – Lots of work customizing Xen, etc., but no proof of integrity
 - ▶ *Guest OS* – Configuration is up to the customer
 - ▶ *Firewall and Storage* – Users must configure their firewall and storage access control is discretionary (also delegate administration)
- *Cloud will be a major target*
 - ▶ *Has the usual problems*

	<<	Jul 20
Amazon Elastic Compute Cloud (API)		✓
Amazon Elastic Compute Cloud (Instances)		✓
Amazon Flexible Payments Service		✓
Amazon Mechanical Turk (Requester)		✓
Amazon Mechanical Turk (Worker)		✓
Amazon SimpleDB		✓
Amazon Simple Storage Service (EU)		✗
Amazon Simple Storage Service (US)		✗
Amazon Simple Queue Service		✗

Integrity-Verified Channels

- Provide an *authenticated, secure communication channel* only when the remote computation satisfies an *integrity criteria*
 - ▶ Thus, the channel establishment verifies the integrity of the distributed system running the computation
 - ▶ *Trust, but verify*



Integrity-Verified Channels

- How might they work? (Think SSL)
- **Setup:** Root Keys
 - ▶ Clients hold *root public keys* necessary to validate *integrity proof public keys* (e.g., TPM AIKs)
 - NOTE: Physical security helps the remote attestation model
- **Negotiation:** On request, get a *proof of distributed system integrity* from the server
 - ▶ Clients validate proof that all components whose integrity is a requirement in a distributed system
 - ▶ Components must track that each satisfies an *integrity criteria*
- **Rekey:** “Rekeying” to revalidate integrity is necessary
 - ▶ Can lose integrity faster than losing a key to an adversary

- **Secure Boot:** Layer N **verifies** layer $N+1$ before loading it -- Does not generate a proof of integrity for remote parties (*AEGIS*)
- **Remote Attestation:** Layer N **measures** layer $N+1$ before loading it resulting in proof of measurements – Does not enforce integrity (*IMA, BIND, NGSCB*)
- **Integrity Model:** Layer N **does not depend** on layer $N+1$ for its integrity – Biba and CW models difficult to achieve in practice
- **Distributed Trust:** Layer N (public key) **asserts** that Layer $N+1$ (another public key) is valid – Does not account for system integrity (*KeyNote, DL, LoA*)

- *Secure Boot*: Trusted software is difficult to come by and validating data is harder, so these systems are very limited
- *Remote Attestation*: Many different mechanisms for measuring and reporting, but all lack comprehensive integrity
- *Integrity Model*: Some exploration into practical, comprehensive integrity models (*CW-Lite*, *UMIP*, *PPI*)
- *Distributed Trust*: Few approaches, but they are underspecified (*BIND*) or lack composability (*Shamon*)

- Applied to the IBM 4758 by Smith (ESORICS 2002, Int J Inf Security 2004)
- *Goals*: Securely boot the 4758 and prove to remote parties (combines secure boot and attestation)
- More specifically, relying party P wants to prove that only entity E holds key K
 - ▶ E is high integrity despite depending on several integrity-relevant events (e.g., boot and upgrade)
- *Defines a precise logic for reasoning about such properties*
 - ▶ But, the 4758 is a very limited system (one application)

Dependency (Integrity)

- Each entity E has a dependency set $D(E)$
 - ▶ An entity E depends on entities that have read/write access to its secrets and write access to its code
 - In general purpose systems, it is primarily dependence on untrusted data that leads to integrity problems
- Implications
 - ▶ Dependency must be comprehensively defined
 - Attestation approaches are incomplete (lack initial integrity, missing events)
 - *Proposal*: Attestation should justify a comprehensive, formal integrity model (CW-Lite, UMIP, PPI)

- History H is a sequence of computation on device
 - ▶ Define the entity E wishes to prove it “owns” K by presenting a $Chain(E, K, H)$ of certificates
 - What is an element of H ? Any integrity-relevant event
 - ▶ Could generate a new certificate for each event in H
- Implications
 - ▶ Large number of integrity-relevant events
 - In 4758, events are few (load one of 4 layers), upgrade layer
 - *Proposal*: Enforce some events to reduce complexity (BIND, Satem, VMV)

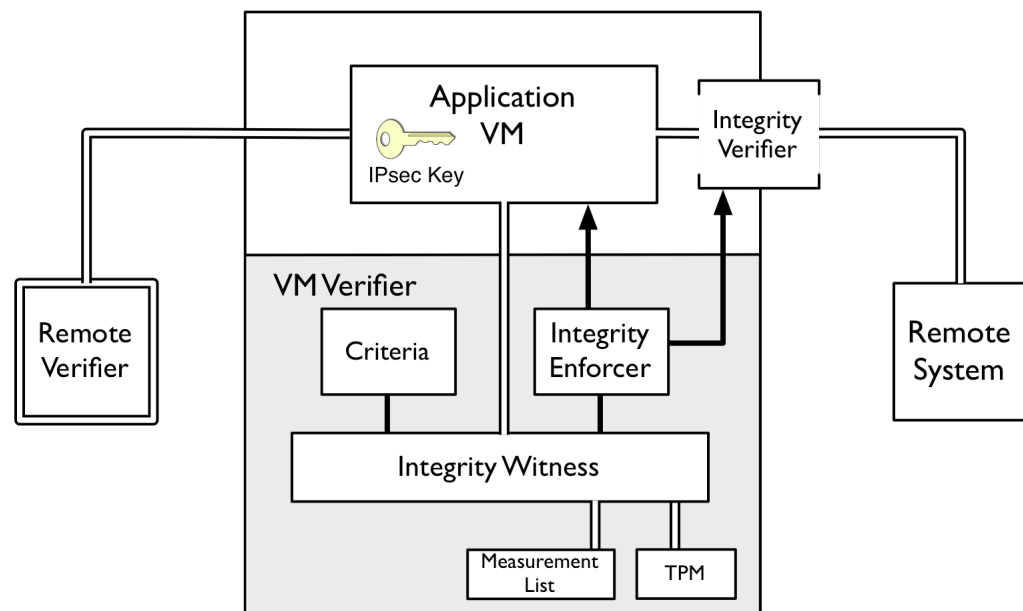
- $D(E) \text{ subset-eq TrustSet}(P) \rightarrow \text{Validate}(P, \text{Chain}(E, K, H))$
 - ▶ OA requires that an entity E's dependencies satisfy the trust set of P to validate that E owns K
 - Difficult for P to determine this for all E and all integrity relevant events
- Implications
 - ▶ P must validate everything
 - So, we don't – we just want a green light (iTurtle)
 - *Proposal*: Use above approach for easy-to-verify base (ROTI) and enforce integrity events against criteria comprehensively

- OA only enables a relying party P to prove that entities in one system own their keys
 - ▶ Want to prove that *all entities in a distributed system* own their keys in the generation of some content C
- Implication
 - ▶ *Proposal*: ROTI-based enforcement can be used for transitive trust (according to criteria)
 - ▶ *Proposal*: Attestation must be generated efficiently (asynchronous attestation)

Technology Challenges

- *Root of integrity* that anyone can verify easily
 - ▶ Base for building and composing proofs
- A meaningful *integrity criteria* that can be proven
 - ▶ Classical integrity, so proofs mean something
- A *comprehensive enforcement mechanism* for integrity
- An *efficient proof generation* mechanism
- A *logic for composition* of proofs
- *Integrity consistency protocols* for keeping channel status consistent with system integrity

- Enforce all integrity-relevant operations
 - ▶ Measure ROTI (VMV) and enforcer's criteria (CW-Lite)
 - ▶ Enable composition of proof verifications from one verifier to another
 - ▶ Maintain state of verifications of system to enable one proof



- Distributed Computing Must Satisfy Security Requirements
 - ▶ *Computing platforms have security mechanisms, but can be misconfigured/compromised and integrity is implicit*
- Integrity-Verified Channel
 - ▶ *A communication channel that justifies the integrity of distributed systems that generate the channel's data*
- We Can Build Such a Channel
 - ▶ *Non-trivial, but the foundations for solving the challenges are emerging*

Questions

- Shamon project
 - ▶ <http://www.cse.psu.edu/~tjaeger/research/shamon.html>
- Penn State SIIS Lab
 - ▶ <http://siis.cse.psu.edu/>
- Email
 - ▶ tjaeger@cse.psu.edu