

# Secure Control of Portable Images in a Virtual Computing Utility

*Ionut Constandache*, Aydan Yumerefendi, Jeff Chase



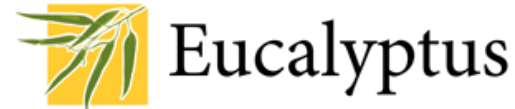
Duke University

# Virtual Computing Playground

- Virtual Machine Technology
  - Server resource sharing
  - Provide computing services computing cloud
- Virtual Machine Images
  - Vehicles for software distribution virtual appliances
- Image Providers
  - Pre-packaged software
  - Static specification:  
**contextualization** step at boot time



Amazon EC2



Virtual Appliance Marketplace



# Problem definition

- Context: Site allocates a VM from a virtual appliance on behalf of an owner.
- Question: How does the owner and the VM instance exchange keys?
  - Preserve independence/orthogonality of the virtual appliance provider and the utility
  - Enable programmatic privileged access as a foothold for **contextualization** (e.g. by a guest controller acting on the owner behalf and which is external to the utility)
- ORCA (Open Resource Control Architecture)
  - Cloud computing/candidate control framework for GENI
  - Allocate **cross-site** resources to client applications (need protocols)

# Contents

- Existing Solutions
- Goals
- Protocols
- Conclusions

# Existing Solutions:



I. Create a Personal AMI (start from a published AMIs)

Solution: The user creates/manages his own VM image

Require:

Technical skills/The user to copy keys inside the VM image

II. Use Public AMIs

Solution: The VM retrieves the keys at boot time from a well-known site-wide location (URL)

Require:

Layer 2/3 security solutions between the VM and the site

Images pre-built with key URL

# Existing Solutions:



Site controls the images, generates the user accounts and passwords



Eucalyptus : Site copies keys before boot

Require:

Site aware of virtual appliance structure, OS, FS, authentication services, keystore locations

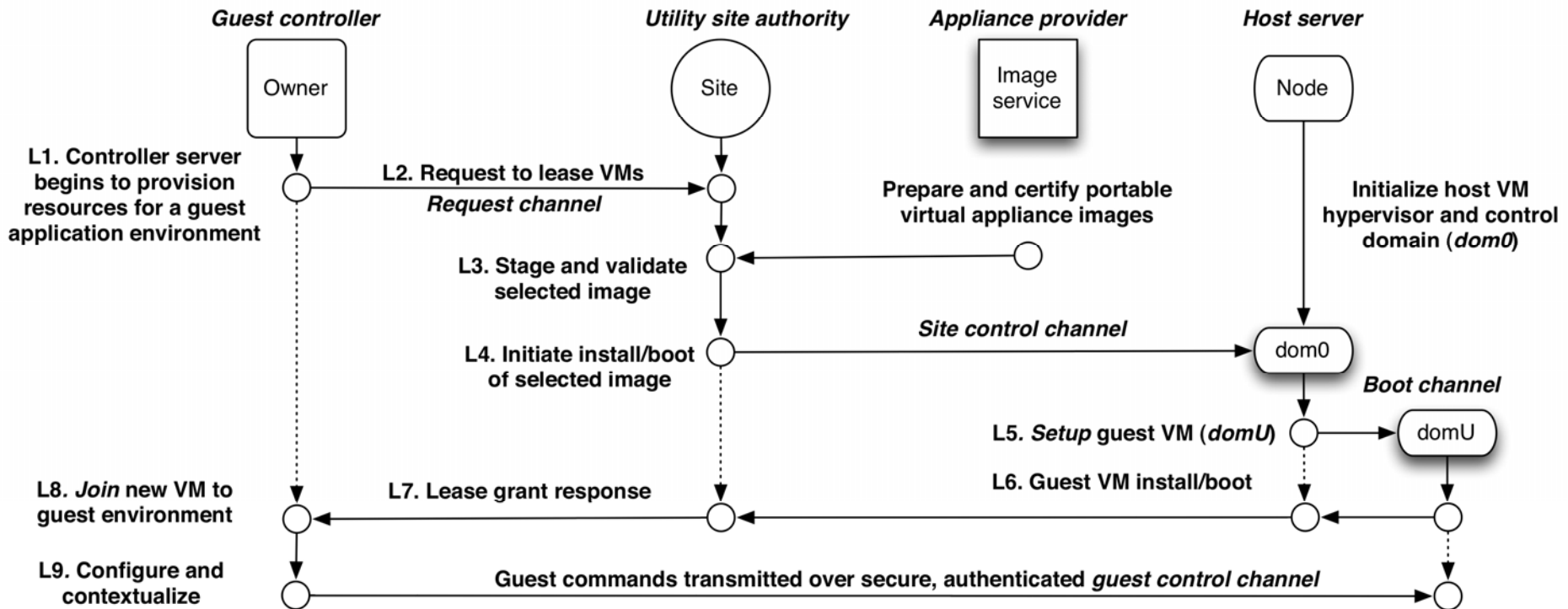
# Drawbacks

- Not a symmetric relation: only the owner gets authenticated to the VM
- Images are not portable
  - Images have site dependencies  
(hard-coded key locations or site-controlled images)
  - Site needs specific knowledge of the image  
(understands the OS, FS, auth services, keystore locations)
  - Images become site specific - inhibits image sharing across sites, support for multiple authentication mechanisms

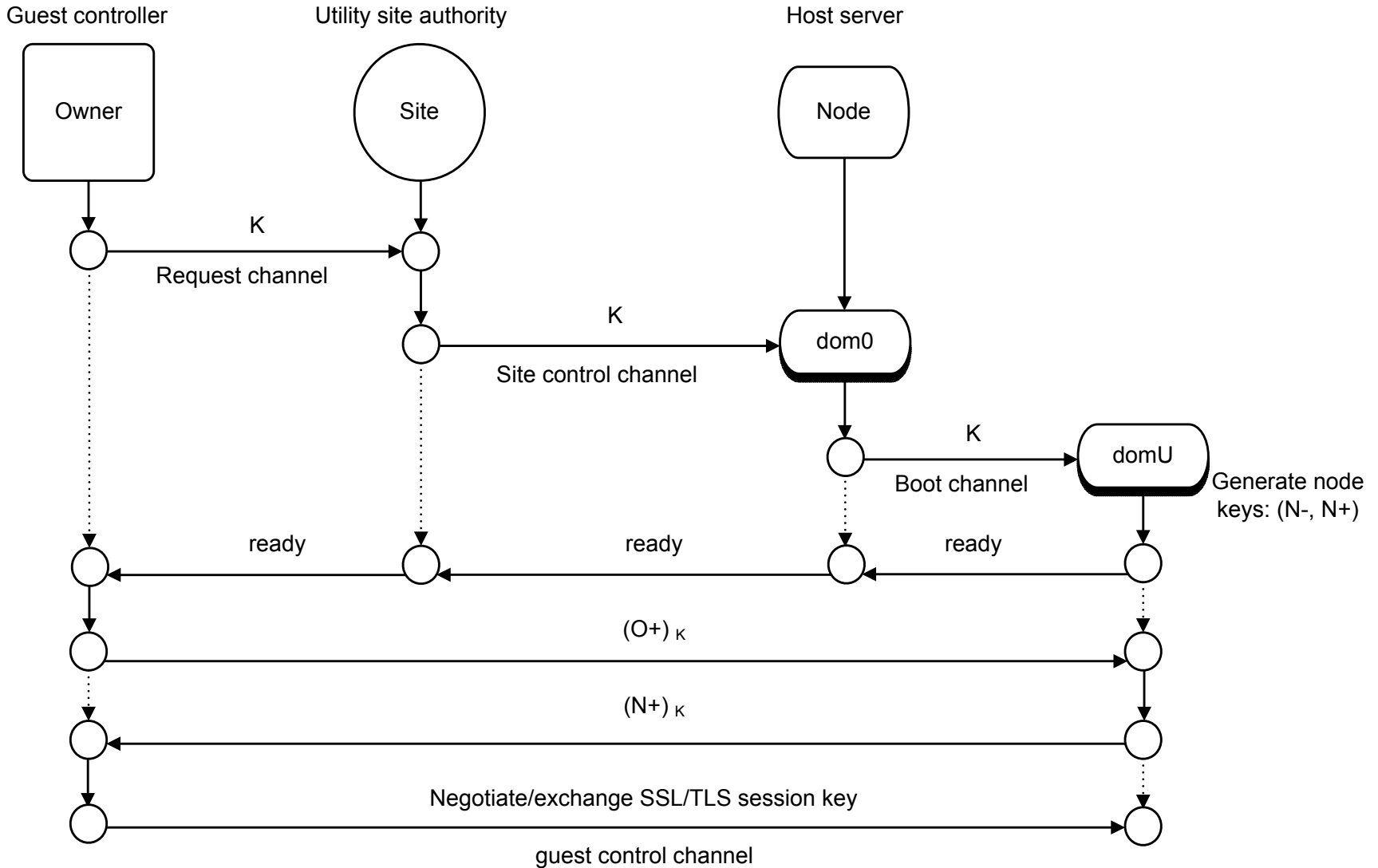
# Goals

- VM control protocol (key exchange)
  - establish a mutually authenticated owner - VM channel at the transport layer
- Preserve image **portability** (virtual appliance)
  - Boot the same VM image at different sites
- Use a common VM control protocol that can support multiple contextualization services
- Make few assumptions about the site/owner's role in the VM control protocol

# VM Boot



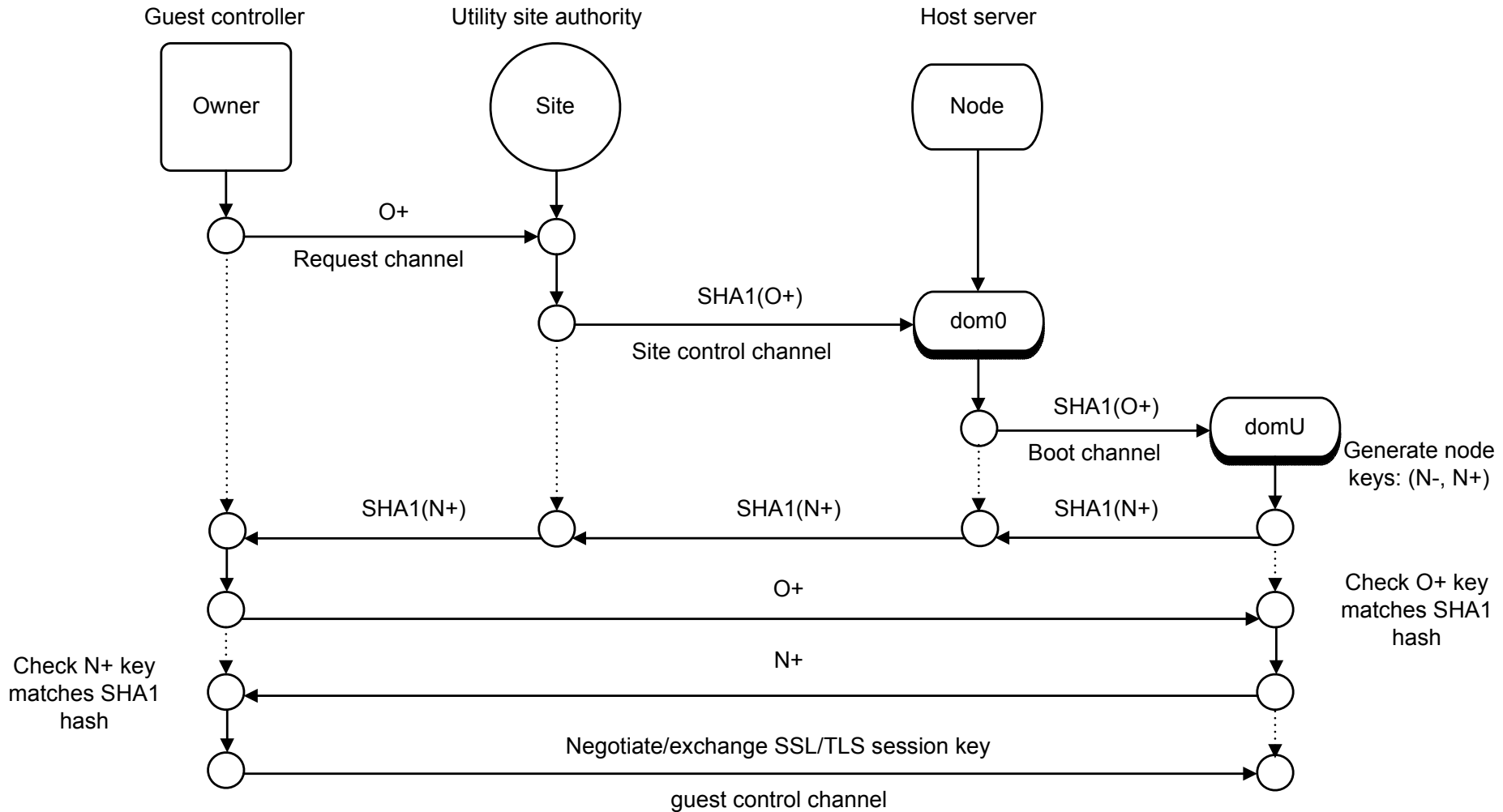
# Shared Secret Protocol



# Shared Secret Protocol

- Defense: man-in-the-middle attack
- Trust: site
  - to pass the security tokens, keep shared secret private
- Secrecy: request, site control, boot channels
- Vulnerabilities:
  - If shared secret compromised: man-in-the middle attack
  - K exposed at the site and at the host server
- Window of Vulnerability: small
  - K used to exchange trusted public keys
- Other: Boot Channel
  - needs small capacity (K~256 bits)

# No "Secrets" Protocol



# No “Secrets” Protocol

- Defense: man-in-the-middle attack
- Trust: site
  - to pass security tokens
- Integrity: request, site control, boot channels
- No vulnerabilities:
  - if site passes the security tokens/not compromised
- Other: Boot Channel
  - small capacity (~160 bits)
  - full duplex

# Conclusions

- Support for portable images/virtual appliances
  - Generic and reusable across sites
  - Easy to manage, share, endorse
- Control Protocol
  - Bootstrap multiple authentication mechanisms (not just *ssh*)
  - Support different contextualization services

Questions?

Thank you!