CDN Architecture Pain Points and ICN Cures?

Christopher A. Wood
UCI and PARC

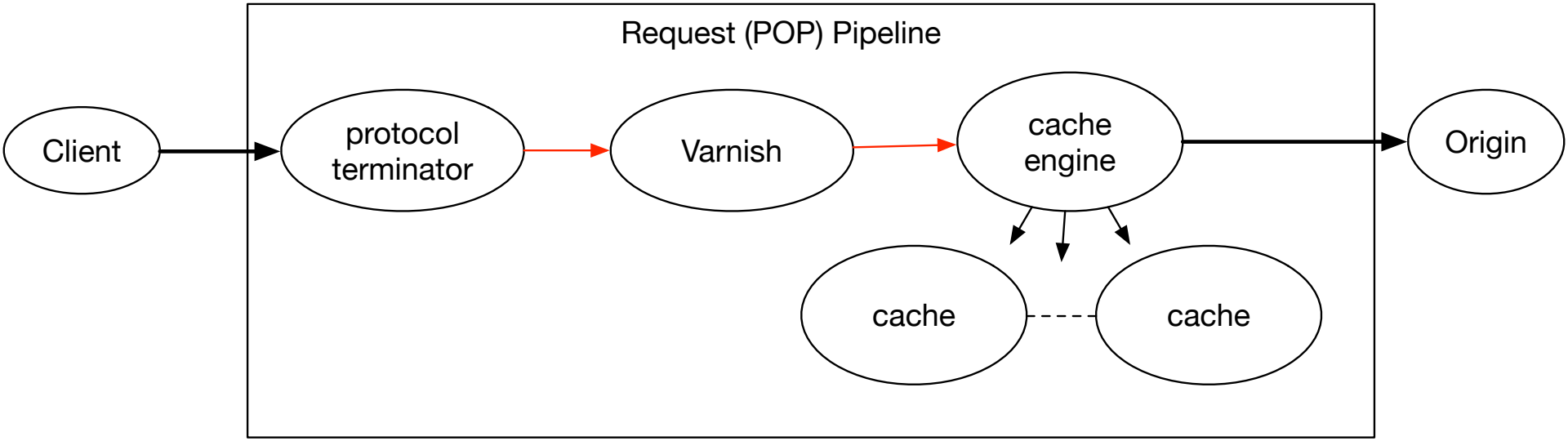
ICNRG Interim Meeting – IETF 96 – Berlin

July 17, 2016

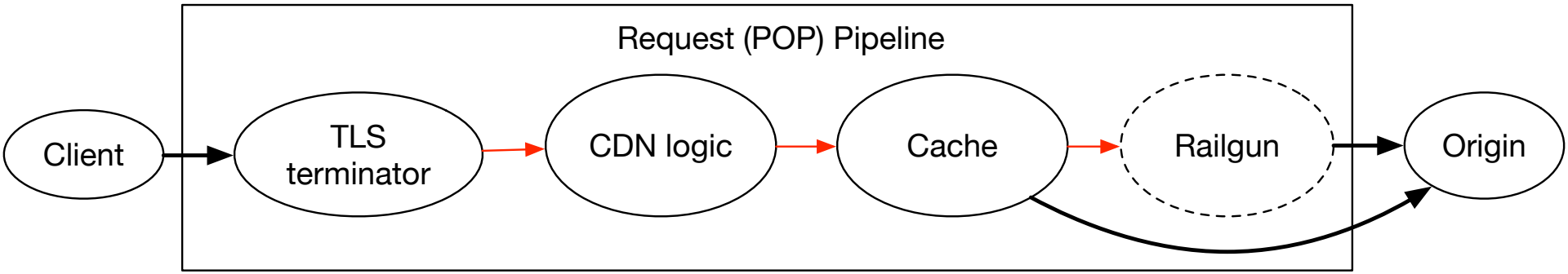
Task

- Talk with people who build and run CDNs
- Explain ICN
- Tease out where it might help

Fastly


Request (POP) Pipeline
Client
protocol terminator
Varnish
cache engine
Origin
cache
cache

CloudFlare


Request (POP) Pipeline
Client
TLS terminator
CDN logic
Cache
Railgun
Origin

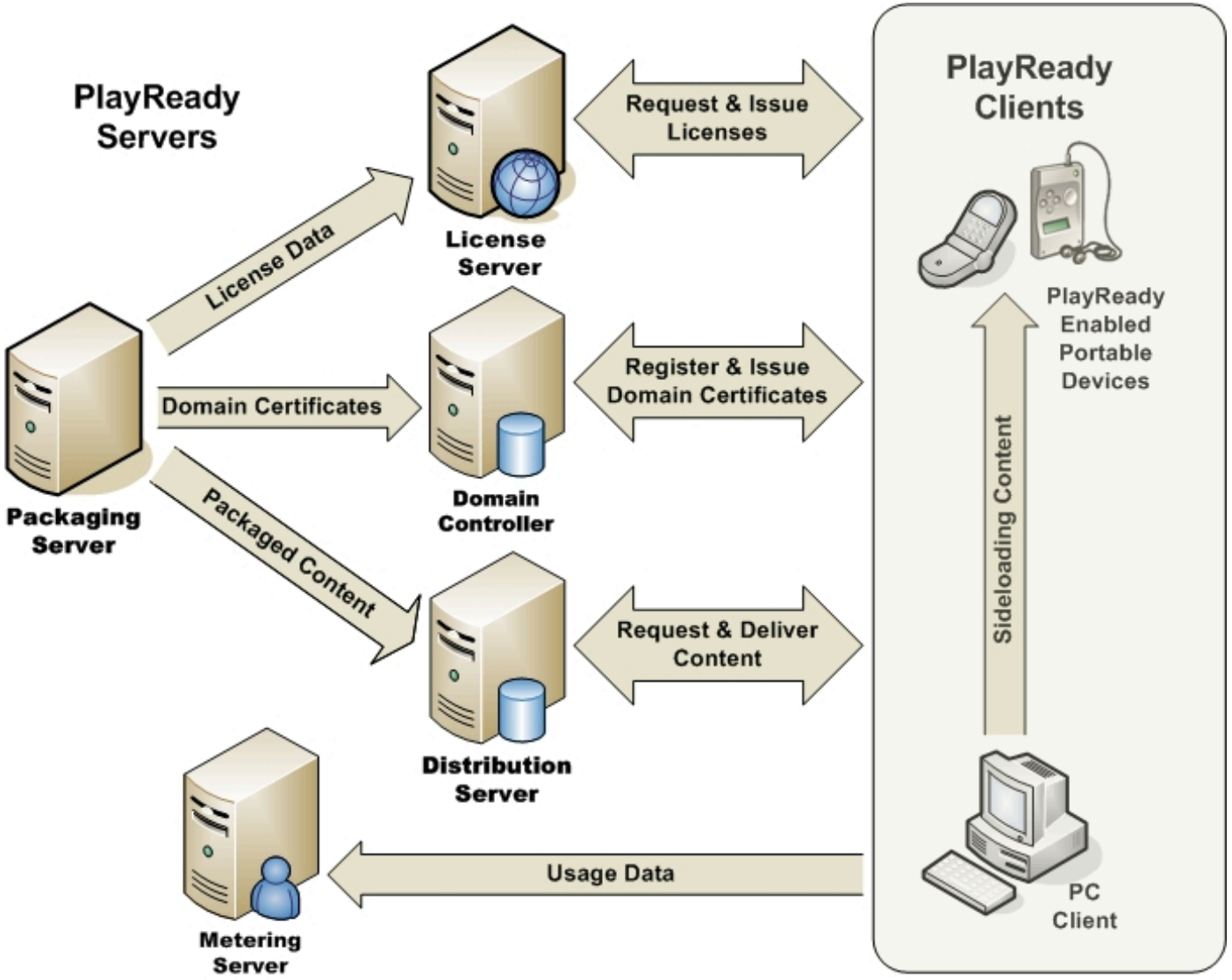
Architecture Patterns

- DNS Anycast for routing-based load balancing
- Edge TLS termination, cleartext internal traffic
  - Keep an eye on LURK
- State synchronization or message passing within POPs
- Push application logic to the edge
  - Treat the origin as a data store or coordinator

TLS Deployment

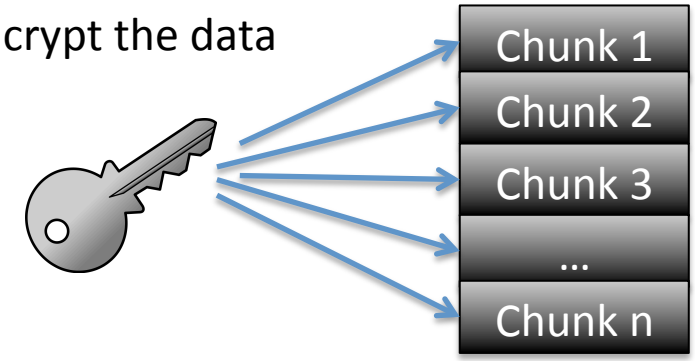
- HTTPS everywhere: **best practice**
  - … but not needed everywhere?
- Is use context-sensitive?
  - EFF: HTTPS everywhere (obviously)
  - Banks, e-commerce, etc.: HTTPS everywhere
  - Netflix: HTTPS for PlayReady manifests and HTTP(S?) for data
    - … for now

Netflix: PlayReady



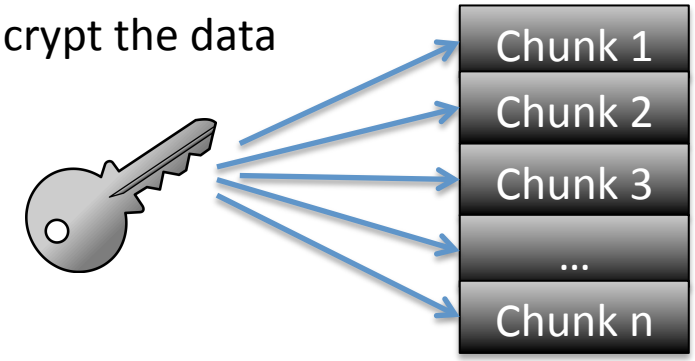
The Netflix Case

1) Encrypt the data

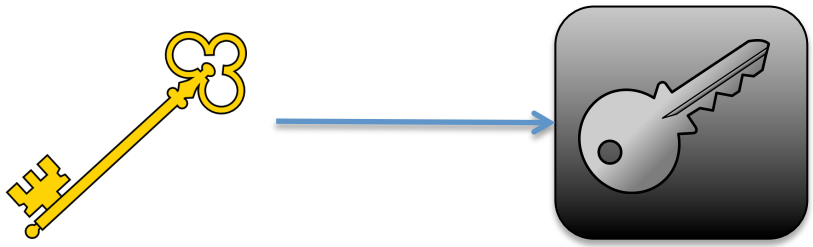


The Netflix Case

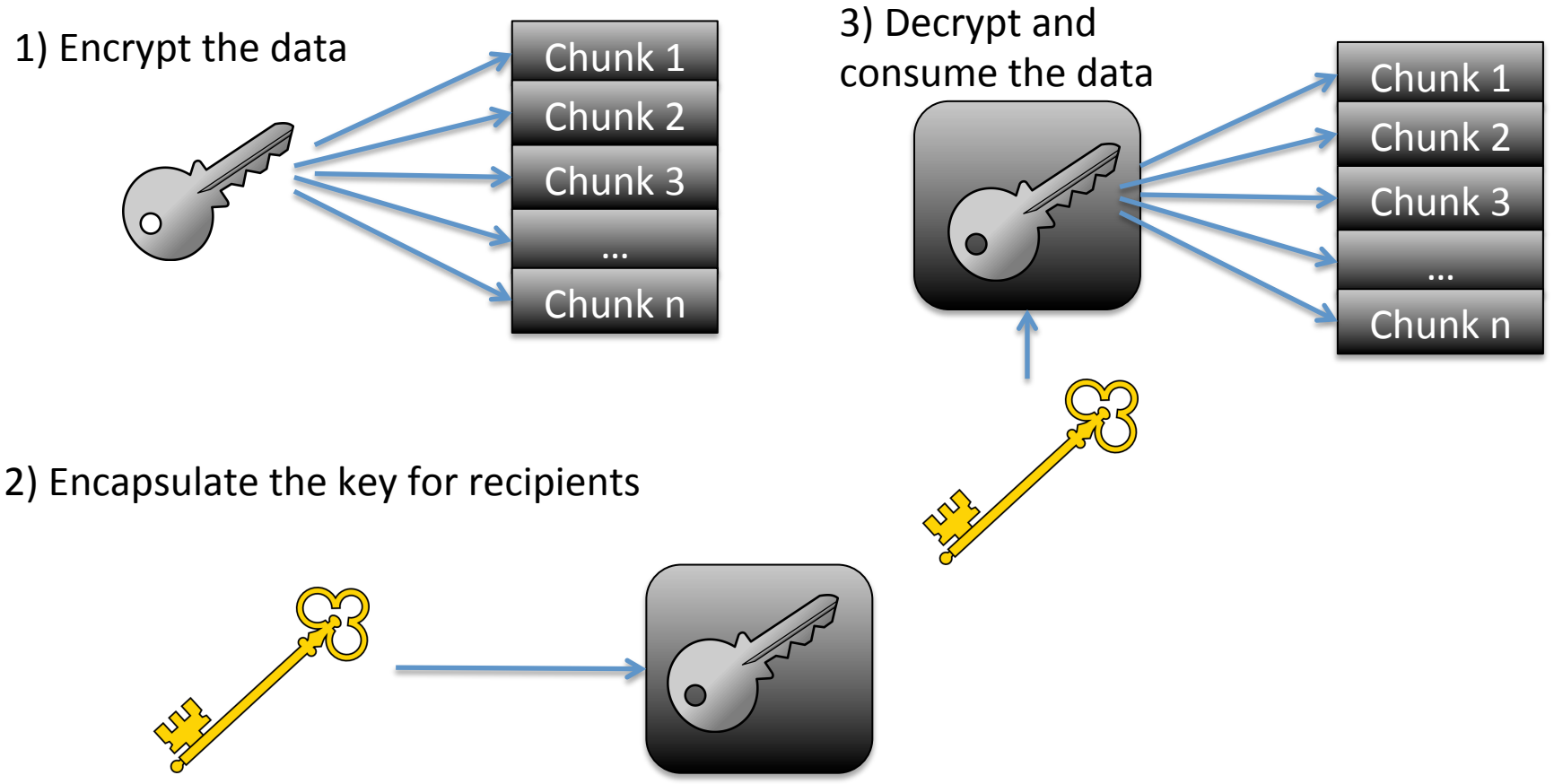
1) Encrypt the data



2) Encapsulate the key for recipients



The Netflix Case



Protection Mechanisms

- Use AES-CBC to encrypt data chunks
- Only authorized consumers can decrypt the PlayReady manifest (license) and obtain the symmetric key
- Rationale?
  - AES-CBC allows for random access
  - Exposure protected by client-specific license key encapsulation

TLS Usage Summary

- Client – CDN and CDN – origin per application request
- If data is already encrypted, it's not necessarily re-encrypted and transferred over TLS

Pain Point #1: DNS Anycast

- Problem: Poor deployment or non-local resolver can result in suboptimal POP selection

Pain Point #2: Tracking State Changes

- Problem: What resources need to be changed when an object is modified?

Pain Point #3: Caching API Requests

- Problem: API requests may be dynamic and the responses typically contain “structured” JSON data

Pain Point #4: Mixed Content

- Problem: Some applications serve HTTP content over HTTPS, or the other way around

Pain Point #5: Event-Driven Content

- Problem: How can we handle “event-driven” content?

Pain Point #6: Distributed Applications

- Problem: Many applications, frameworks, etc. are not engineering with caching in mind

Pain Point #7: TLS Termination

- Problem: How do CDNs and origin servers coordinate to share private keys without causing long-term problems?