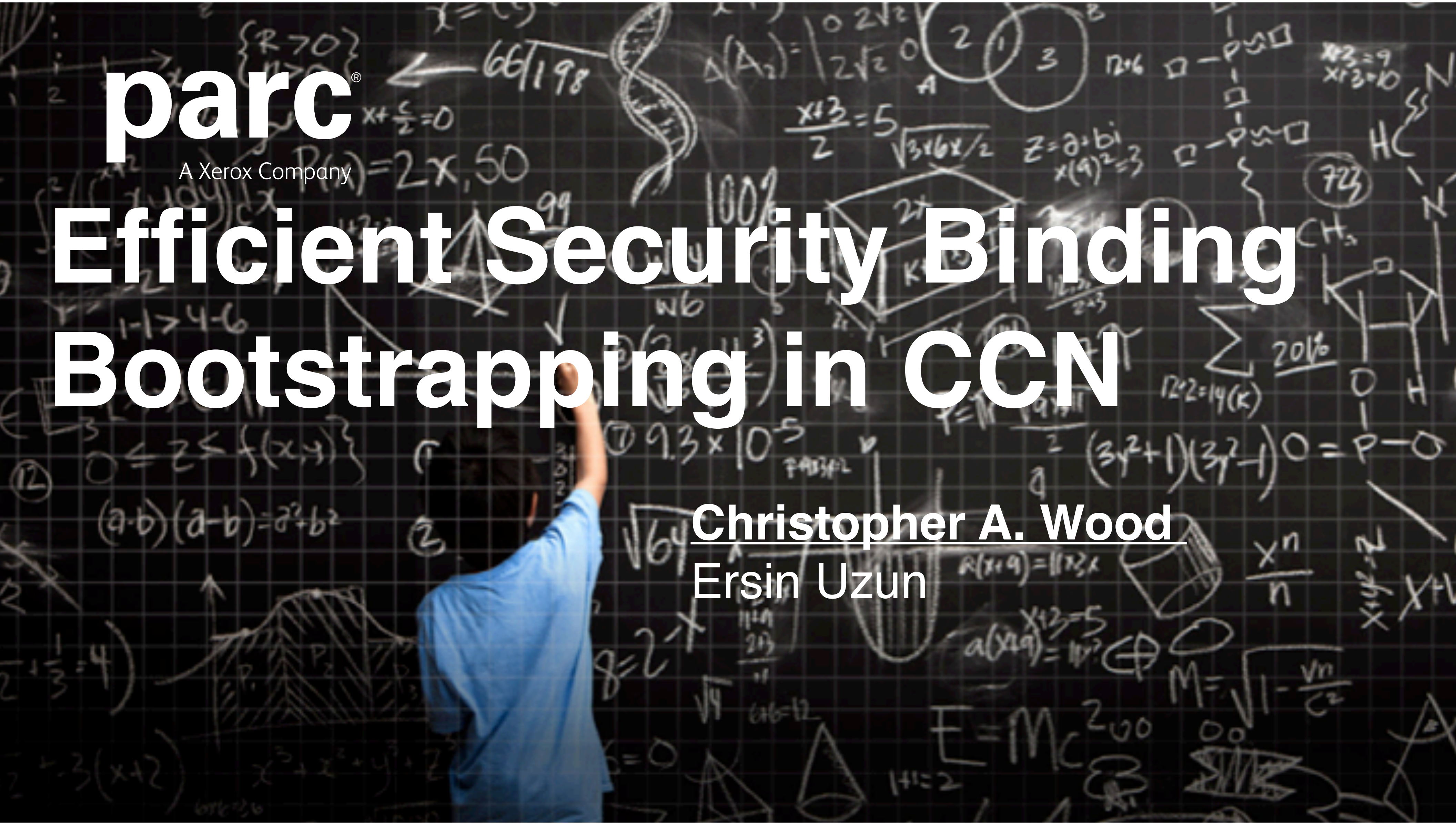


Efficient Security Binding Bootstrapping in CCN

Christopher A. Wood
Ersin Uzun

Agenda

1. Content Object security and signature verification overview
2. Verification bottlenecks and possible optimizations
3. Different data formats
4. Different retrieval methods
5. Conclusion

CCN Security Overview

Assumption: Classical PKI infrastructure is used for assigning, issuing, and trusting cryptographic keys

Security Basics:

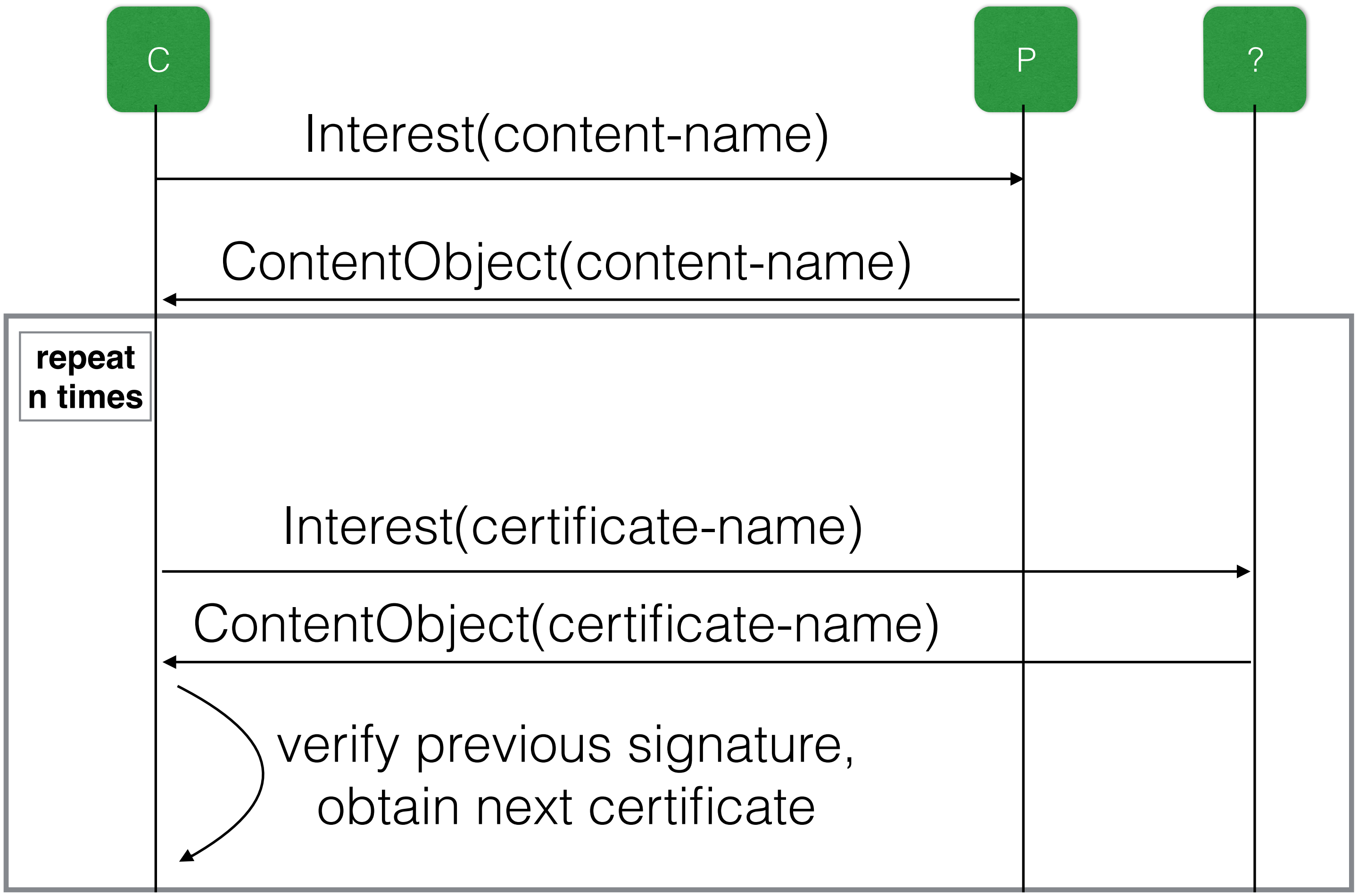
Content Objects are verified by consumers and routers

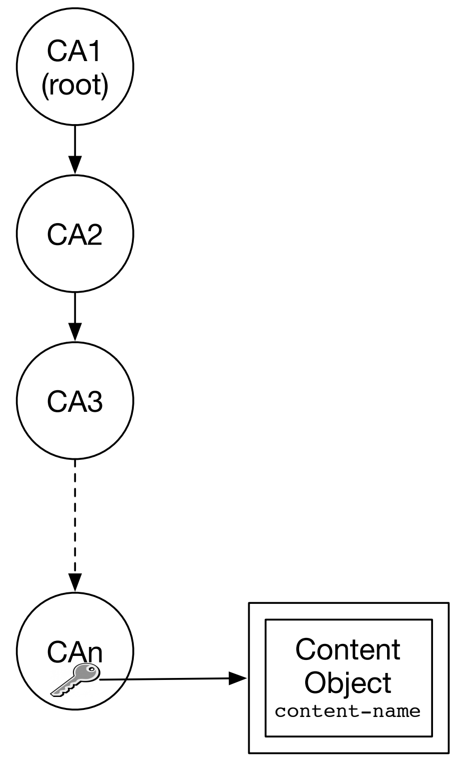
- Public verification keys and certificates are Content Objects

- Trust of keys and certificates is rooted in anchors

Recall: verification is more efficient if the content hash is known a priori

Content Verification Process

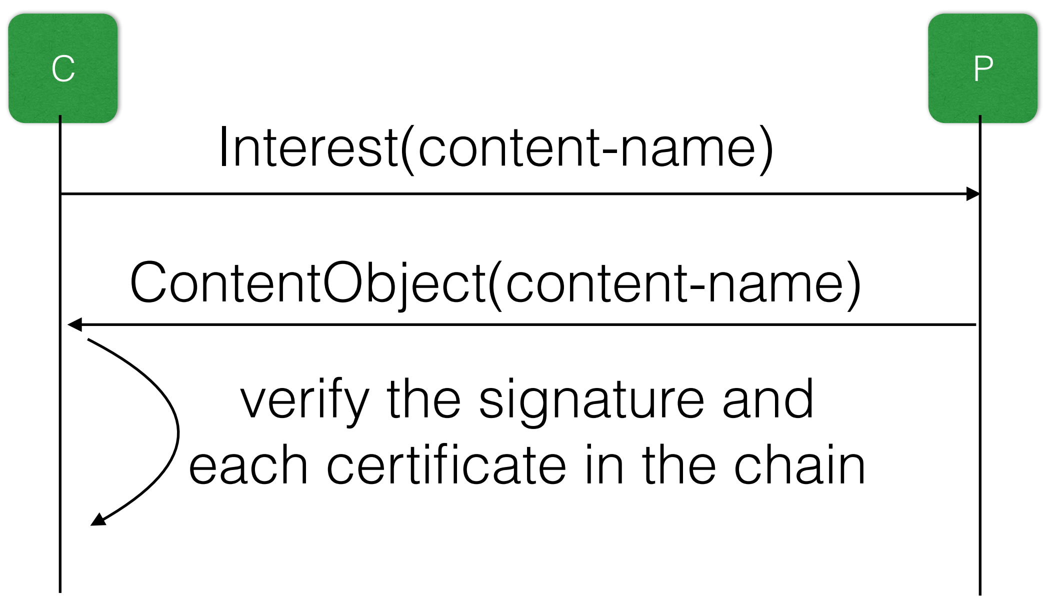


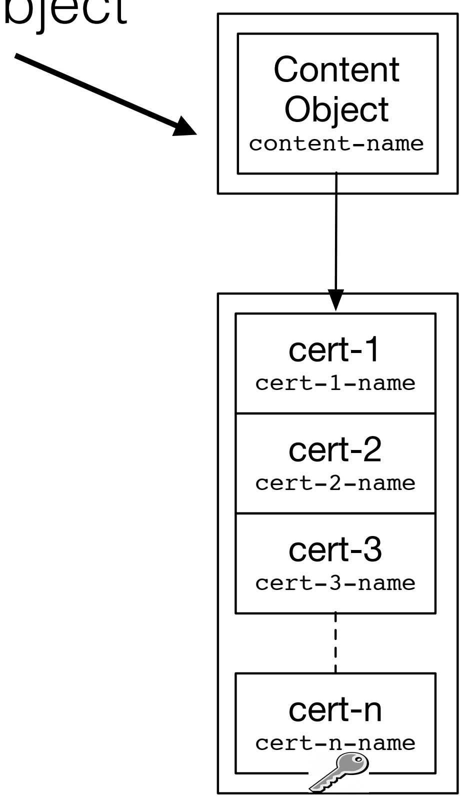


parc
A Xerox Company

A More Plausible Scenario…

Embed or link to the entire chain from the Content Object





Performance Bottlenecks

Hash-based content retrieval is very efficient:

- No public key cryptographic operations or key retrieval issues

Bootstrapping hash-based retrieval induces performance issues:

- Message complexity (key resolution and certificate chain traversal)
- Bandwidth complexity (certificate size)
- Computation and time complexity (linear in chain length)

Optimization Dimensions

Bottlenecks	Potential Optimizations
Message & Bandwidth Complexity	Modify certificate retrieval
Computation Overhead	Modify certificate format

parc A Xerox Company

Certificate Data Format

The overhead induced by public-key certificate data formats could be reduced via:

(1) Implicit certificates

(2) Aggregate signatures

Implicit Certificates

Traditional certificates are composed of three parts:

1. Identification data
2. Public key
3. Digital signature from certifying authority

Implicit certificates combine parts **(2)** and **(3)** into the **same value**.

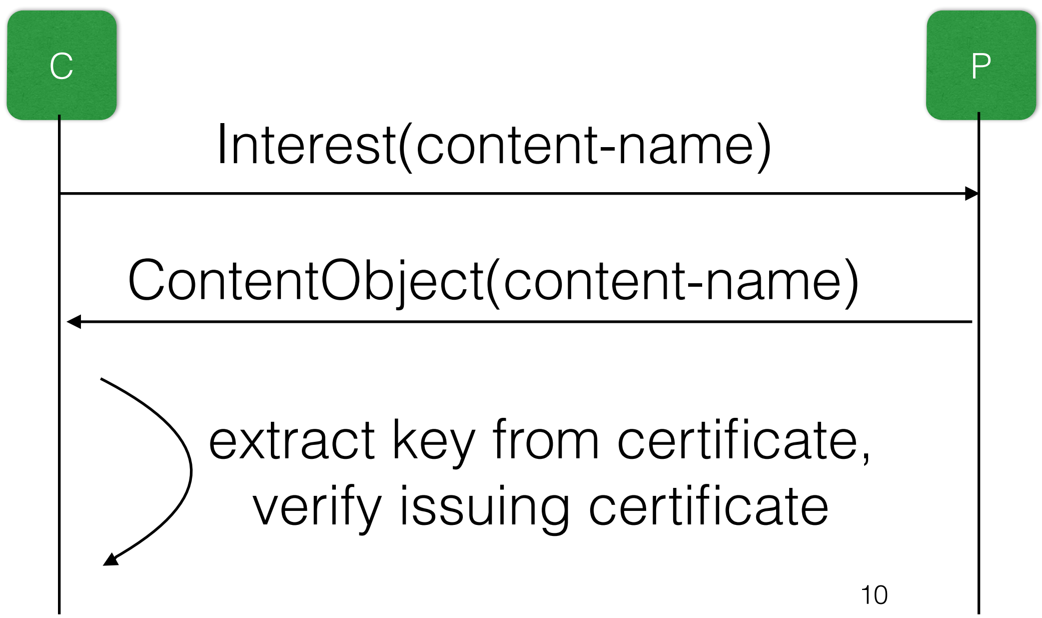
Successful key extraction from the certificate implicitly verifies the signature

parc A Xerox Company

Implicit Certificates (cont’d)

Proposal: Embed (now smaller) certificates with Content Objects

Instead of **verifying n signatures**, attempt to **extract n public keys**



Cost(verification) < Cost(extraction)

Drawbacks

1) Certificate chains cannot be longer than 3 nodes

2) Potential denial-of-service attacks

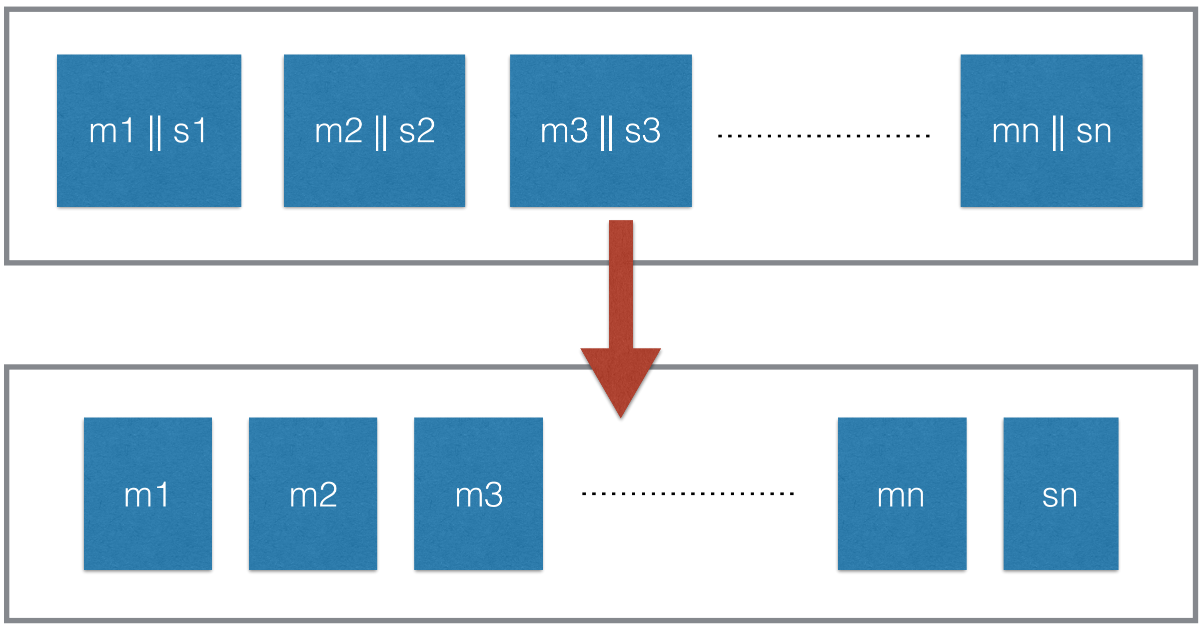
3) Requires composition with ECDSA to be useful and secure

4) … and more.

See http://www.secg.org/sec4-1.0.pdf for more details.

Aggregate Signatures

Aggregate signatures combine **n signatures over n distinct messages** into a single signature that can be **verified at once**



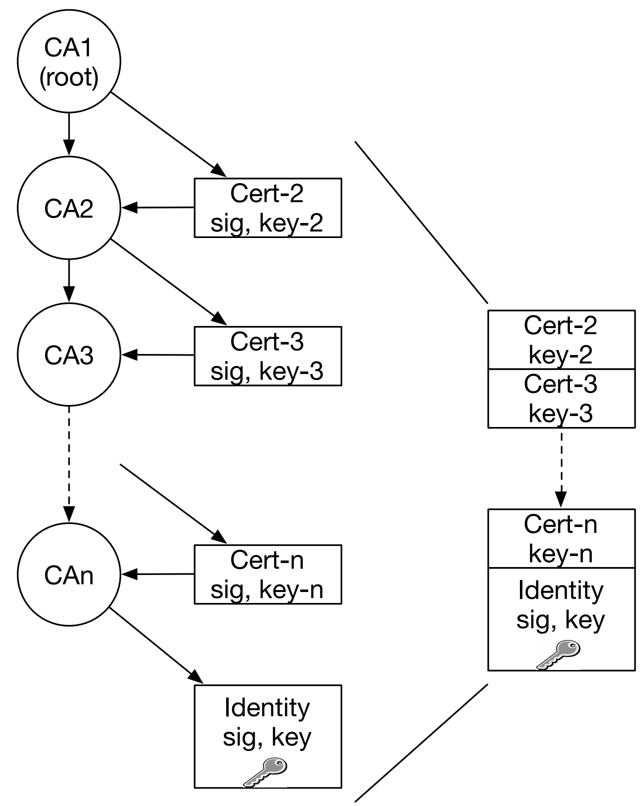
parc
A Xerox Company

Compressed Certificate Chains

Recall: Hierarchical trust based on certificate chains are used for content objects

Goal: Compress certificate chains with **n signatures** in a chain to **a single signature**

Content objects can link to compact chains



Observations

Other fancy cryptographic techniques could be applied with other limiting bottlenecks

Current implicit certificate and aggregate signature schemes require cheaper or fewer cryptographic operations

… but, these computational savings **do not justify** their use

Observations

Other fancy cryptographic techniques could be applied with other limiting bottlenecks:

Current implicit certificate and aggregate signature schemes require cheaper or fewer cryptographic operations

… but, these computational savings **do not justify** their use

Claim: reducing message and bandwidth complexity is more important

Certificate Retrieval Optimizations

Certificates are likely to be retrieved on-demand for each content object (i.e., not embedded with content objects, but **linked**)

Goal: Reduce the number of requests for certificates

Key Catalogs

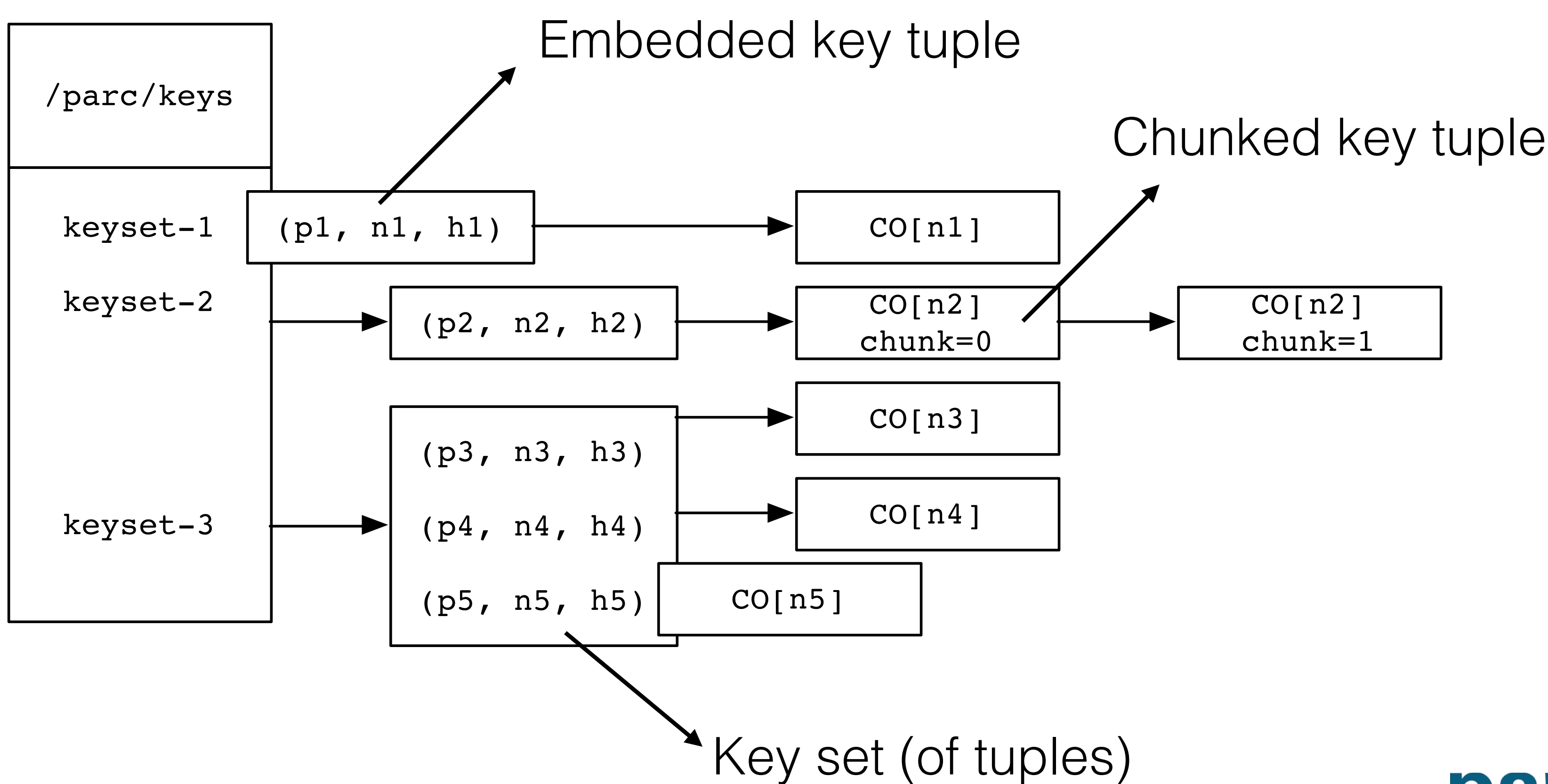
Producers can build and produce a Manifest-based catalog of keys

Each entry is a [name(prefix), key(name), hash] tuple

Consumers fetch the key catalog for a producer, e.g., during "registration"

Verify the key catalog signature once and then efficiently access all other keys

Key Catalog Structure



parc A Xerox Company

Encoding

```
KeyCatalog      = [KeySetLink]
KeySetLink      = <CCNx Link> | KeySet
KeySet          = [KeyTuple]
KeyTuple        = KeyTupleIndex KeyTupleValue
KeyTupleIndex   = OCTET
KeyTupleValue   = <CCNx Name> <CCNx Name> HASH
HASH            = 32(OCTET)
```

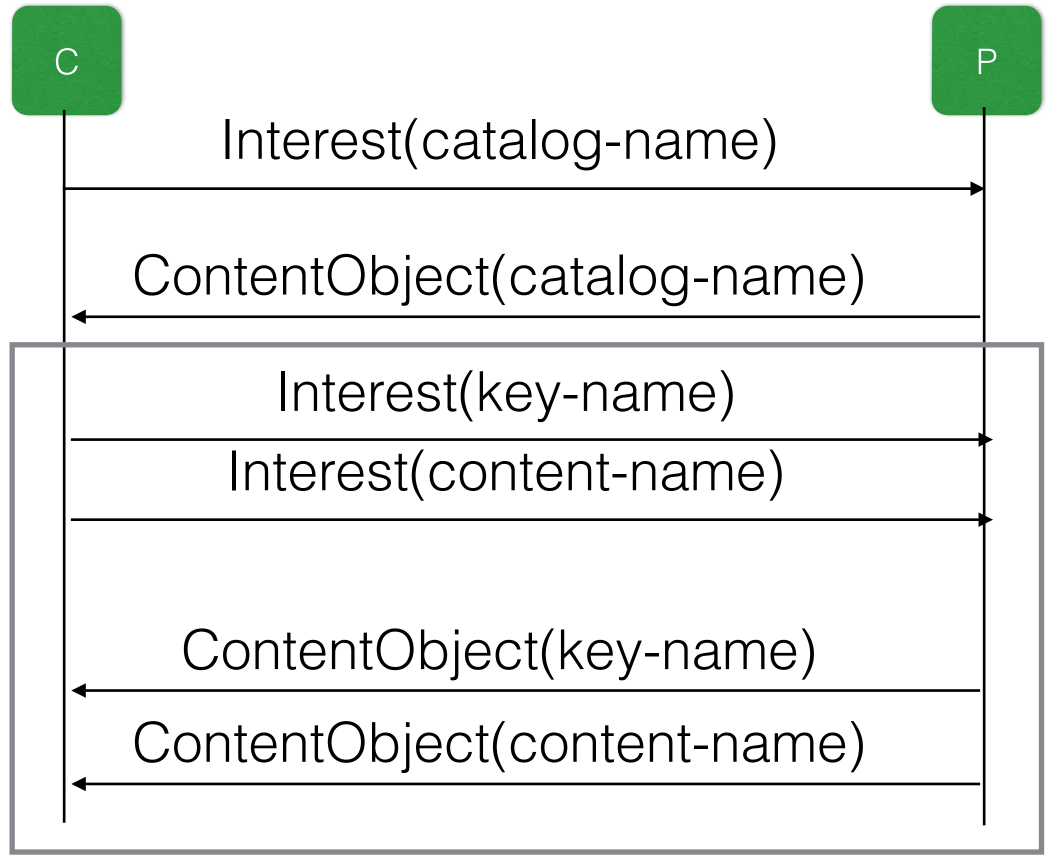
This can be mapped to a CCN Manifest encoding

Key Catalog Usage

Possible use cases:

1) All future keys can be fetched in parallel with content objects

2) All producer keys can be prefetched and stored



parc
A Xerox Company

Catalog Discovery

Consumers must know or discover the key catalog name:

- Installed with application software
- Inferred from a well-known name for each application, e.g.,

```
lci:/parc/csl/nds/key-catalog
```

- Provide a link to the key catalog in each content object

Catalog Discovery

Consumers must know or discover the key catalog name:

- Installed with application software

- Inferred from a well-known name for each application, e.g.,

`lci:/parc/csl/nds/key-catalog`

- Provide a link to the key catalog in each content object

Content objects can point to related keys or key catalogs

Wrapping Up

Discussed verification problems and performance bottlenecks

Applications are better served by optimizing the retrieval of certificates

Use Manifest-based key catalogs to efficiently access certificates

Questions?...

Thank you

parc®

A Xerox Company