What information centric networking can do for peer-to-peer

IRTF – P2PRG WG

July 2010

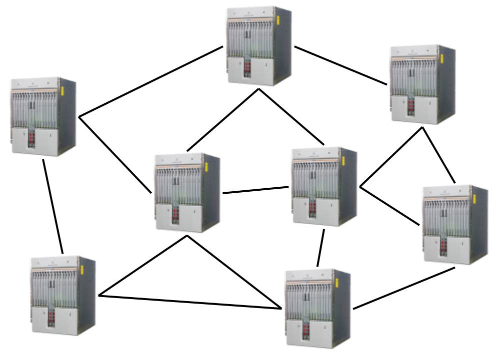
Ove Strandberg & Börje Ohlman

Motivation: Information centric networking

Today's Internet

Conversations between Hosts

Host-centric abstraction



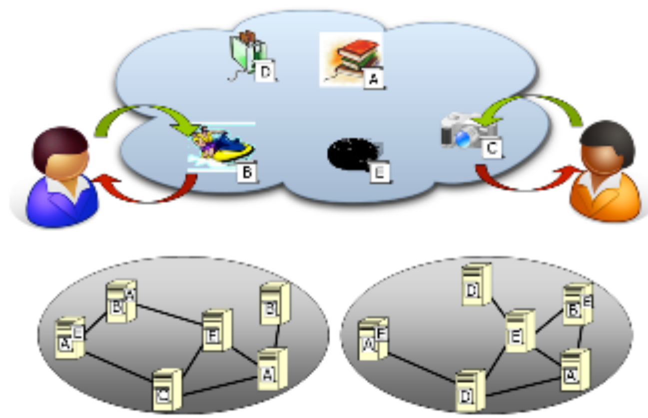
Evolution

Future

Information-centric Network

Dissemination of Information Objects

Information-centric abstraction



- No common *persistent naming scheme* for Information
- Security is host-centric
  - Mainly based on *securing channels* and *trusting servers*
  - Can't trust a copy received from an untrusted server

Outline

- P2P ID challenges
- Secure Naming Structure
  - PPSP draft
- Peers in the network infra – Network storage
  - Decade
- Secure naming & P2P application interaction
- Metadata as additional data identifiers
  - future P2P application features?
- Prototyping and validation of NetInf

P2P data identification challenges

- Identification of the same data at different location require knowledge of multiple data IDs (host centric addressing)
- Streaming application have their own identification system
  - Hard to use same data between different p2p application
- Security based on trusting hosts
  - selection of arbitrary source is not possible as only trusted hosts can be used

Secure naming characteristics

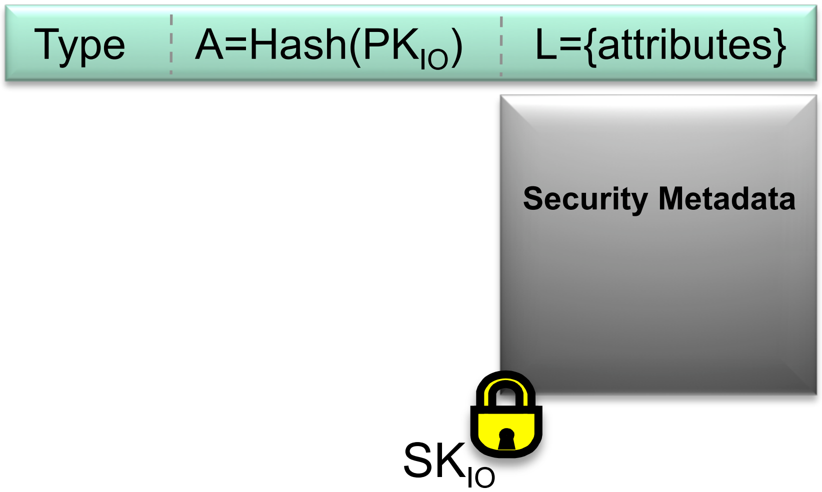
- Self certified ID
  - using hash of data
- Name persistence, in spite of
  - Location changes
  - Content changes
  - Owner changes
  - Organizational changes

Self-Certification

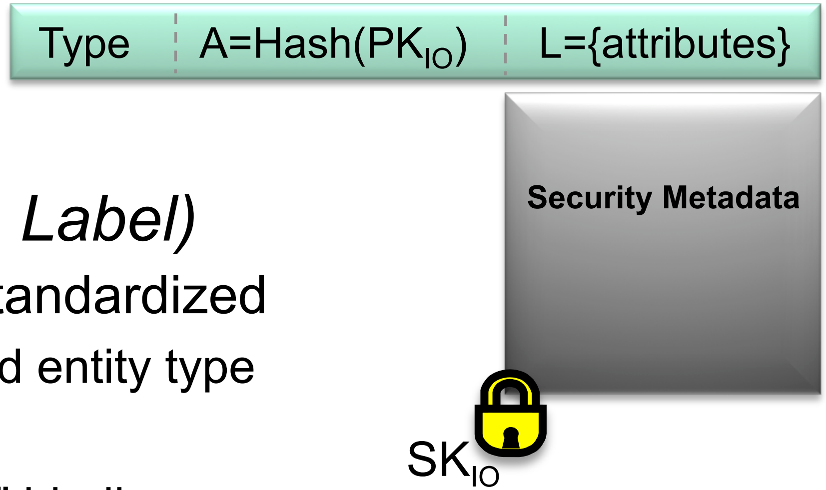
- Prevent unauthorized changes, ensure data integrity
  - Important to support data retrieval from any available copy/source
- Static content
  - Include *hash(content)* in ID *Label* field
  - Advantage: no need to retrieve metadata
  - Verification: compute *hash(retrieved data)* and compare to hash in ID
- Dynamic content
  - Storing *hash(dyn.content)* in ID would violate ID persistence
  - Store *hash(content)* in security metadata and sign with SK_{IO}
  - Verification:
    - Verify that signature is correct and corresponds to PK_{IO}
    - Compute *hash(retrieved data)* and compare to hash in security metadata

Naming Scheme Overview 1

- Information Object (IO) = (ID, Data, Metadata)
- Each IO has an *owner*
- All equivalent copies have the same ID
  - This might include different versions



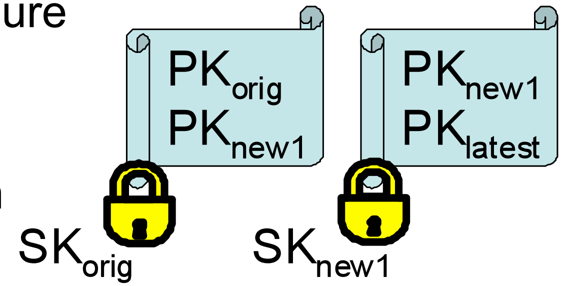
Naming Scheme Overview 2



- ID = (*Type tag*, *Authenticator*, *Label)*
  - *Type* tag: mandatory, globally standardized
    - Adapt naming scheme to named entity type
  - *Authenticator A*: bind ID to PK_{IO}
    - Secure “ID – security metadata” binding
    - (Original) owner authentication (see owner change)
  - *Label L*: Arbitrary, ensure global uniqueness
- *Security metadata*
  - All information required for embedded NetInf security features
  - Securely bound to ID via PK_{IO}/SK_{IO} pair

Name Persistence

- Location change
  - Based on ID/locator split
  - ID dynamically bound to network location(s) via name resolution service
- Content change
  - See self-certification
- Owner change
  - PK_{IO}/SK_{IO} pair conceptually bound to IO, not owner
  - Basic approach: PK_{IO}/SK_{IO} pair securely passed on to new owner
    - Disadvantage: not robust with respect to SK disclosure
  - Adv. approach: new owner uses new *PK'/SK'* pair
    - Sign metadata using the new *PK'/SK'* pair
    - Securely bind *PK'/SK'* pair to ID via certificate chain



- Owner's organizational change
  - IDs are flat and do not reflect organizational structures

Owner Authentication and Identification

- Owner authentication separated from data self-certification
  - By allowing the corresponding PK/SK pairs to be different
  - Owner authentication is possible even if multiple owners use the same PK/SK pair for data self-certification
  - More freedom in the choice of PK/SK pairs for data self-certification
- *Owner authentication* binds self-certified data to owner's PK
  - Include hashed owner's PK in self-certified data and sign this data with the corresponding SK (anonymous)
  - Build up trust in (anonymous) owner by reusing PK for different IOs
- *Owner identification*: in addition, bind self-certified data to owner's real world identity
  - Achieved like owner authentication, where owner's PK and identity data are included in self-certified data
  - Owner's PK and identity are bound by PK certificate issued by TTP

Network storage - Decade

- The offline and badly connected peers problem is mitigated by in-network storage
  - Week uplink networks
- Content caches are easily migrated towards flash-crowds
- Capacity challenges with too much localized traffic from peers (ALTO) can be mitigated by network storage

Secure naming & P2P application interaction

- With self-certifying names, the data received is the data requested in P2P system
- In today's P2P system, no guarantee that the downloaded content actually matches the expected/correct content
  - Like forged torrent file and/or data file can be inserted
- Additions to P2P
  - Extend torrent file with additional security metadata
  - Generate torrent name along draft method (draft-dannewitz-ppsp-secure-naming-00.txt)

Metadata

- Secure naming structure supports additional metadata
  - Needed for instance for PK_D and signing purposes, persistent naming
  - Additional metadata can be data attributes:
    - Classification
    - Meaning of data
    - Data status
- Search
  - Metadata can be used for attribute based search
  - Potentially more accurate search than full text search

Evaluation

- Java-based NetInf prototype
- Naming scheme proved easy to implement
  - Based on established security mechanisms (encryption, digital sign.)
- Easy to integrate and use naming scheme in applications
  - Built applications from scratch
  - Extended existing applications (e.g., Firefox, Thunderbird)
- Example: Firefox plugin
  - Interprets links containing NetInf IDs instead of URLs
  - User adv.: automatic content integrity check, reduce broken links
  - Publishers adv.: simplify content management via persistent IDs
- Load and overhead not an issue
  - Implementation also smoothly running on Android cell phones

NetInf Prototype

- Implementation includes
  - Self-certification
  - Persistent IDs
  - Owner authentication
  - Basics of owner identification
- Algorithm used
  - Can use any encryption/signature algorithm.
  - Currently use RSA and SHA1 for the hashing

Summary and Conclusion

- Tracker, network storage and peer relevant issues
- Information-centric type of networks have inherent need for secure naming scheme
- Secure naming structure combines features not available in existing naming schemes
- Example of torrent changes
- Feasibility of secure naming demonstrated via prototyping:
  - http://www.4ward-project.eu/
  - http://www.netinf.org

Thank you for your attention

Background slide