Naming and Security

NDN Services for Tactical Networks Tutorial

milcom

Military Communications for the 21st Century

November 12–14, 2019 • Norfolk, VA, USA

Defining Multi-Domain Command and Control

Security in File Transfer

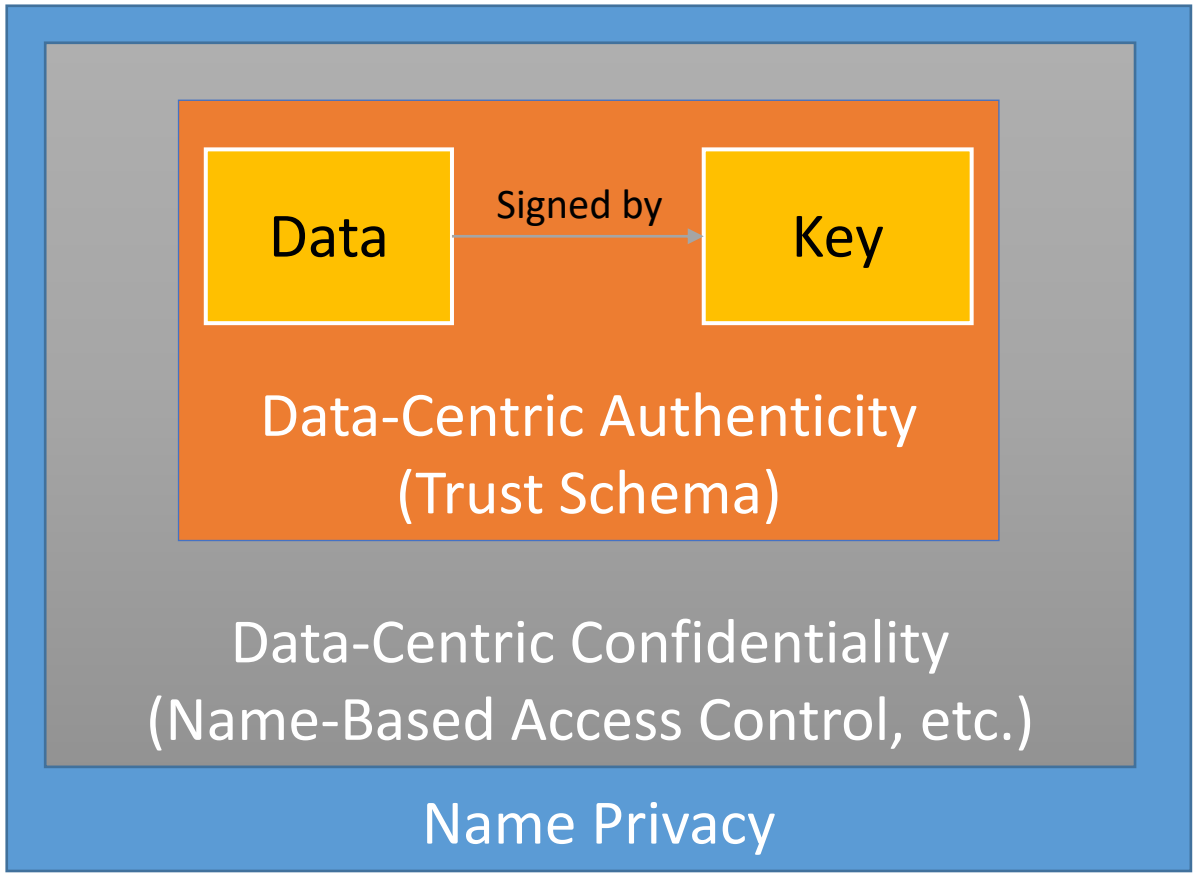
milcom
Military Communications for the 21st Century
November 12-14, 2019 • Norfolk, VA, USA
Defining Multi-Domain Command and Control

- We have file transfer application that by definition needs to have
  - files must come only from the right people (authentication)
  - files can be read only by authorized people (confidentiality)
  - names of the files (and general context) should be revealed only to authorized people (name confidentiality)
- All of these in the face
  - multiparty file transfer (multiple publishers of different files, multiple consumers for each of the file
  - connectivity could be disrupted at any time
  - communication may resume over alternative channels

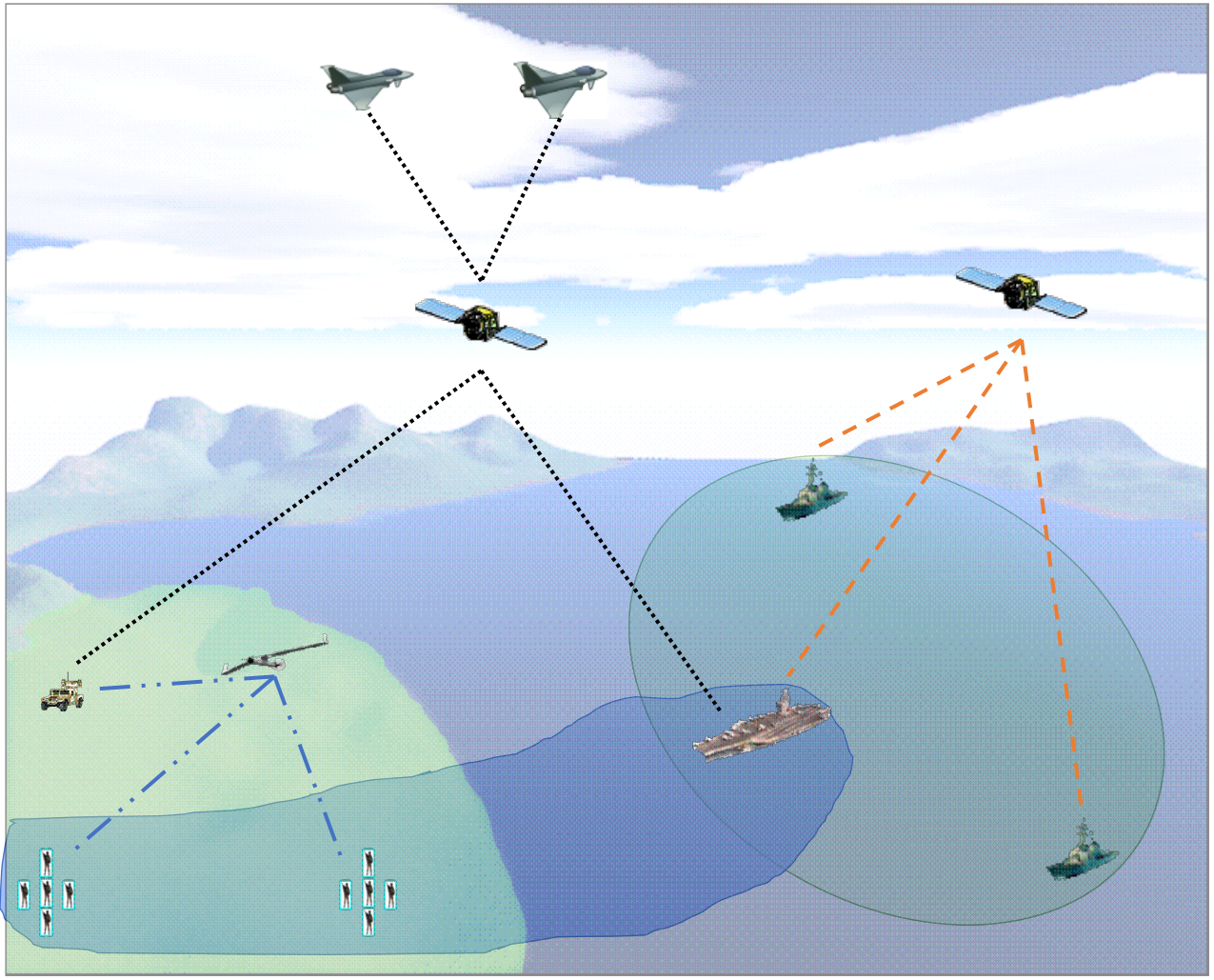
Required Security Properties

- Mitigate false data packet injection
  - built-in data authentication
  - automated key and policy management leveraging NDN naming
- Mitigate eavesdropping
  - optional data encryption (for content confidentiality)
  - optional name encryption (for name privacy)
  - automated key and policy management leveraging NDN naming

NDN Security



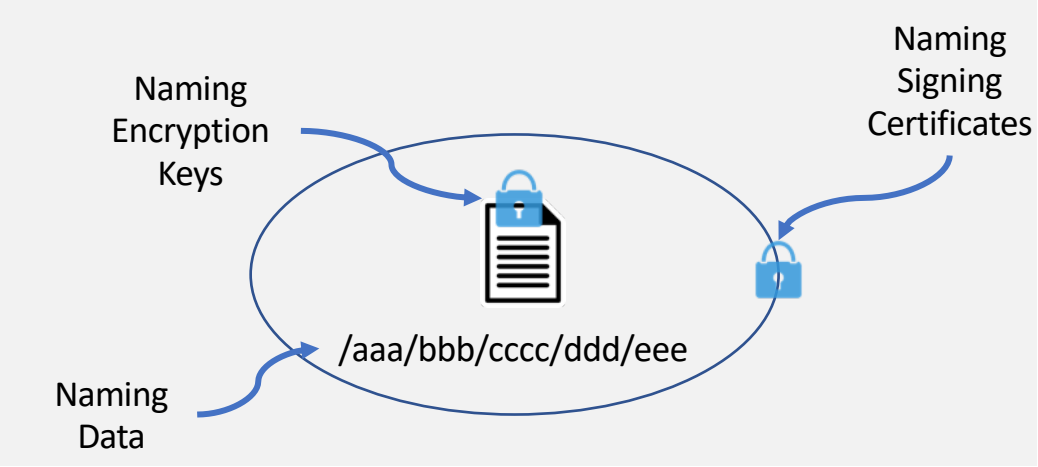
Notional Tactical Network



- File transfer among the troops, ships, and aircrafts
- Lossy environment
- May have significant delays
- Multiple delivery channels

Naming in a File Transfer App

- Naming is part of the application design and configuration
- What to name
  - Files
  - Signing keys
    - and policies
  - Encryption keys
    - and policies
- Application cares about
  - fetching the data from those who are authorized to participate within a given interest group



Naming Files

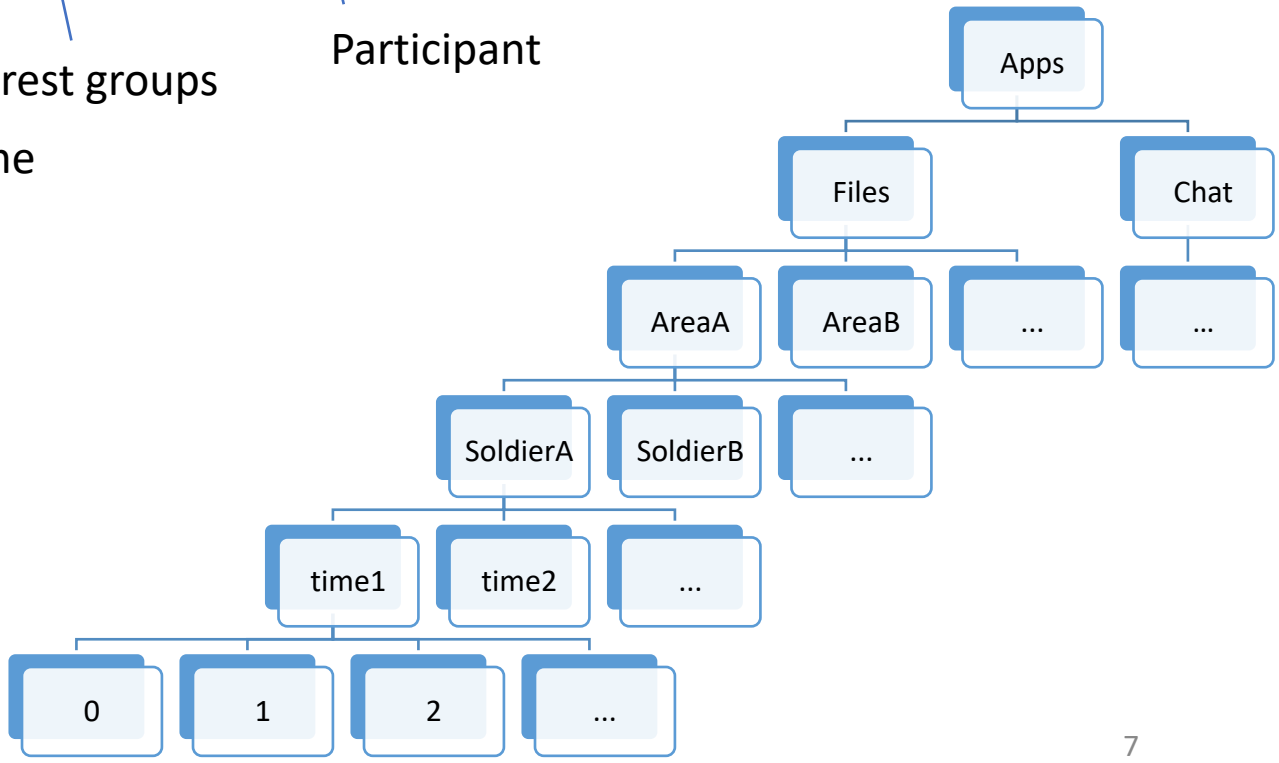
/Apps/Files/AreaA/SoldierA/timestamp/segment/...

Prefix

Application name

Interest groups

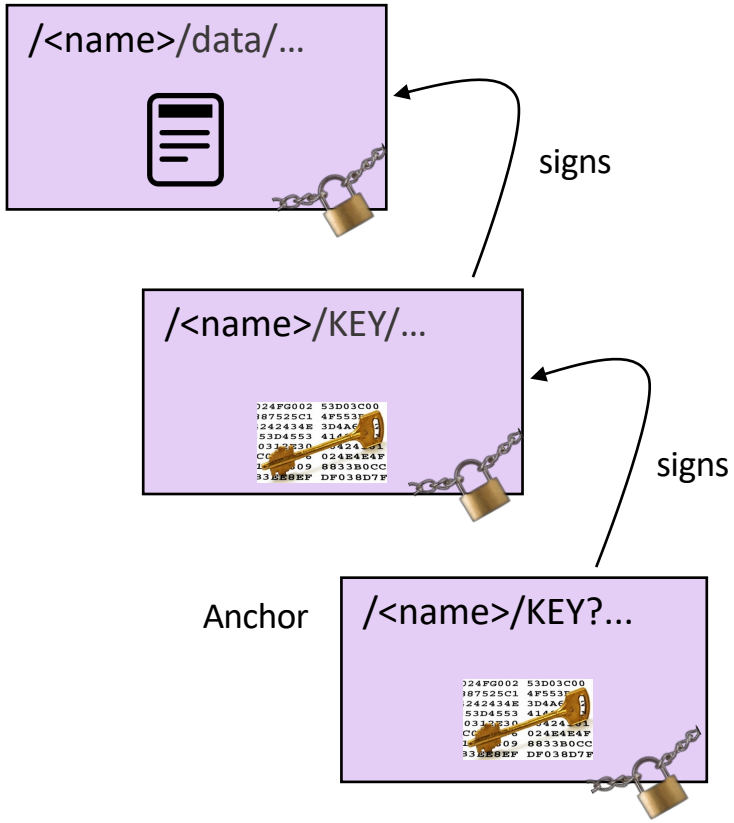
Participant



Data Authentication

Built-in Data Authentication

- Each piece of data (a file segment)
  - is signed using crypto keys
- Key itself is a data packet
  - is named and signed by a “next level” key
- The data-key-key-key-…-root key chain
  - validates integrity and authenticity of every single piece of data
  - regardless how it arrived on the system



Naming Signing Keys

/Apps/Files/AreaA/SoliderA/KEY/<id>/…

/Apps/Files/SoliderA/KEY/<id>/…
/Apps /SoliderA/KEY/<id>/…

User keys

"Mission" keys

/Apps/Files/ AreaA/KEY/<id>/…
/Apps/ AreaA/KEY/<id>/…

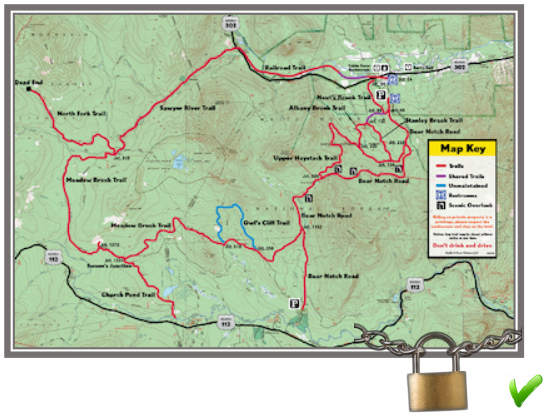
/Apps/ KEY/<id>/…

Base key (anchor)

Not Just Signature, but Whose Key Signed It?

milcom
Military Communications for the 21st Century
November 12-14, 2019 • Norfolk, VA, USA
Defining Multi-Domain Command and Control

/Apps/Files/AreaA/SoldierA/timestamp/segment/…



/Apps/Files/AreaA/SoliderA/KEY/<id>/…

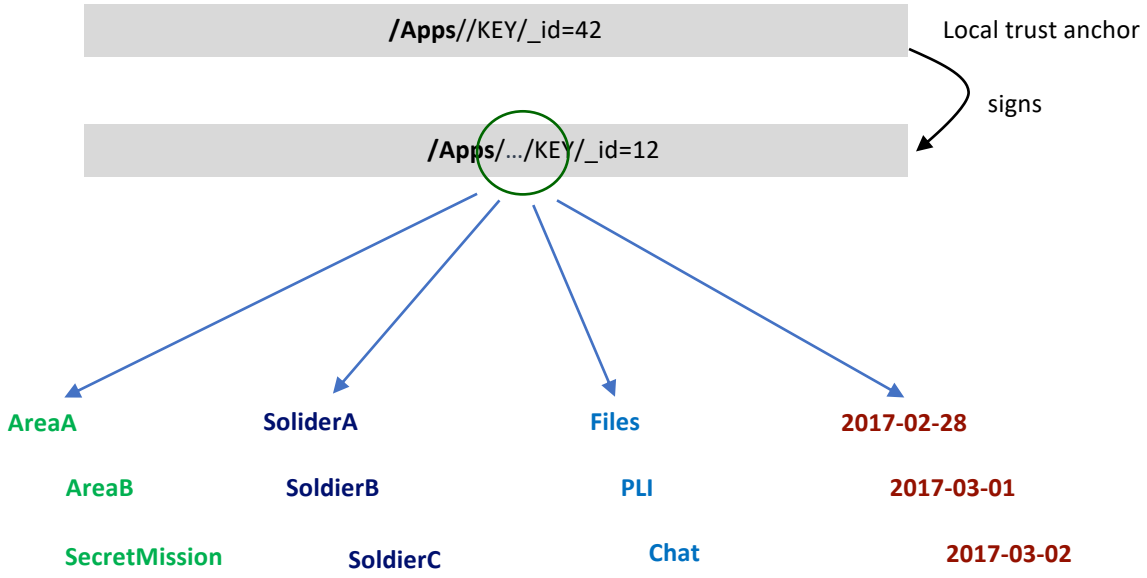
A valid files segment published by a soldier in a mission

/Apps/Files/AreaA/SoldierA/timestamp/segment/…

/Heisenberg/KEY

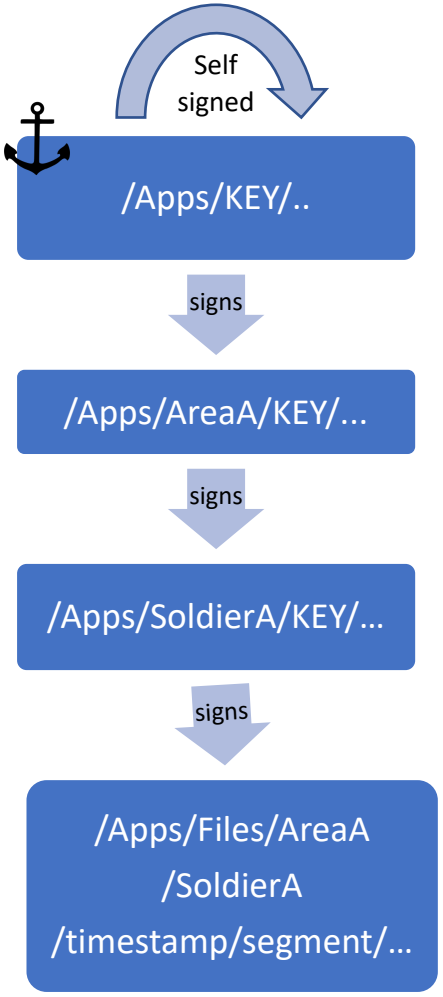
A forged file segment

Defining Limits via Namespace Design



Relation of Data and Keys

- /Apps as the shared trust anchor
  - Cert name /Apps/KEY/<id>/…
  - Securely installed out-of-band into all user devices
- Base creates mission keys signed by the trust anchor
  - /Apps/AreaA/KEY/<id>/…
- Everybody has a key (cert) signed by the mission key
  - /Apps/SoliderA/KEY/<id>/…
    - A user creates a file transfer app key to sign data
      - /Apps/Files/SoliderA/KEY/<id>/…
- File segments signed by app-user (mission) key



Signing Policy: Trust Schema

milcom
Military Communications for the 21st Century
November 12-14, 2019 • Norfolk, VA, USA
Defining Multi-Domain Command and Control

/Files/Policy/_v1

File segments

(:Prefix:<>*)<Files>(:Mission:<>)(:User:<>)<><>…

/Apps/**Files**/AreaA/SoldierA/timestamp/segment/…

User key

(:Prefix:<>*)(:User:<>*)<KEY>[key-id]

/Apps/SoliderA/KEY/<id>/…

Mission key

(:Prefix:<>*)(:Mission:<>*)<KEY>[key-id]

/Apps/AreaA/KEY/<id>/…

General Trust Model

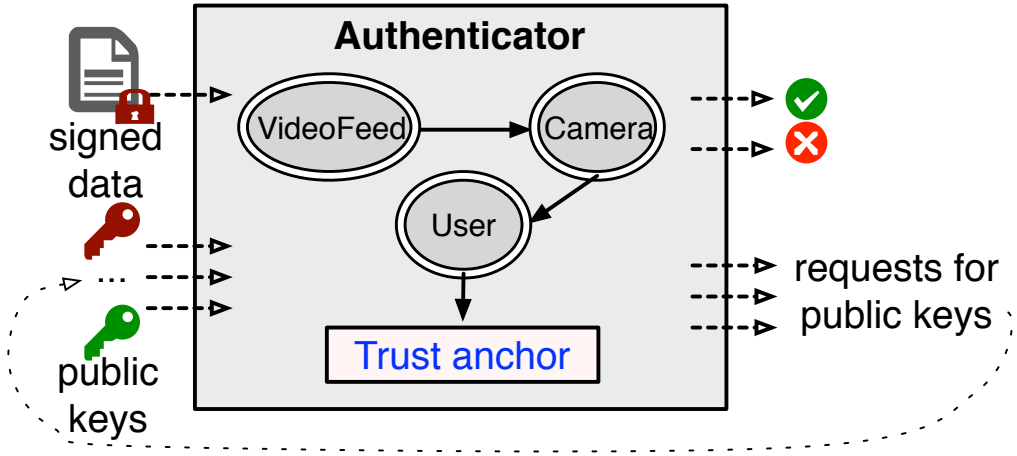
Anchor key

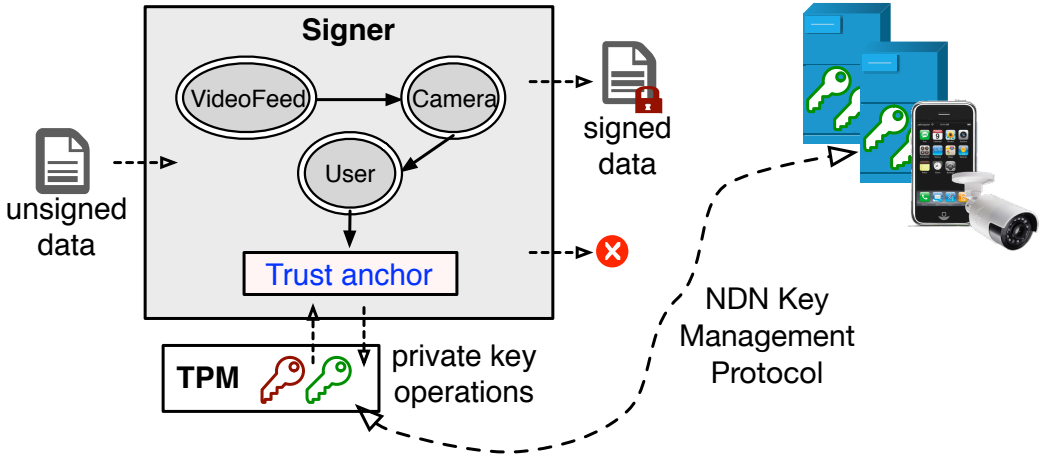
/Apps/KEY/_v5

Trust Model Specialization for the App

Trust Schema as an Automation Tool

milcom
Military Communications for the 21st Century
November 12-14, 2019 • Norfolk, VA, USA
Defining Multi-Domain Command and Control

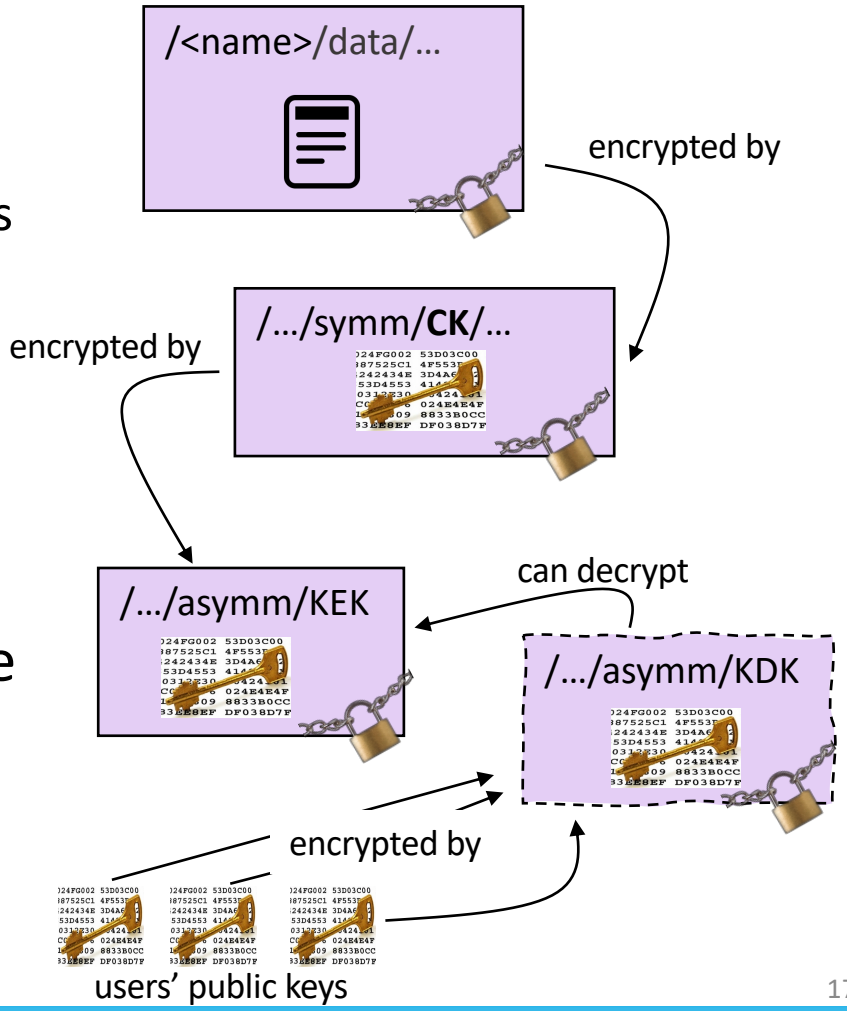




Data Confidentiality

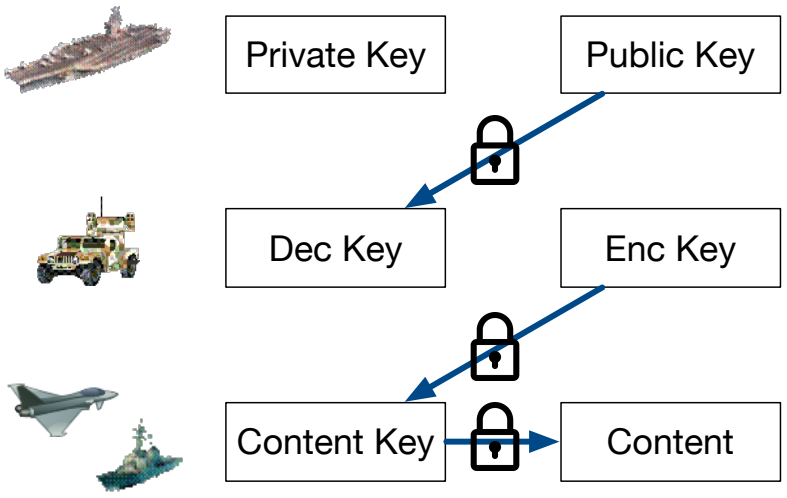
Data Confidentiality via Encryption

- Each piece of data is encrypted
  - symmetric key to encrypt content (CK)
    - 1 key for group of packets (segments of a file)
  - asymmetric key to encrypt encryption key (KEK/KDK)
    - 1 key per access group
    - name can define granularity
  - asymmetric keys (or other mechanisms) to secretly distribute decryption key
    - provisioning mechanism to decrypt symmetric keys and content

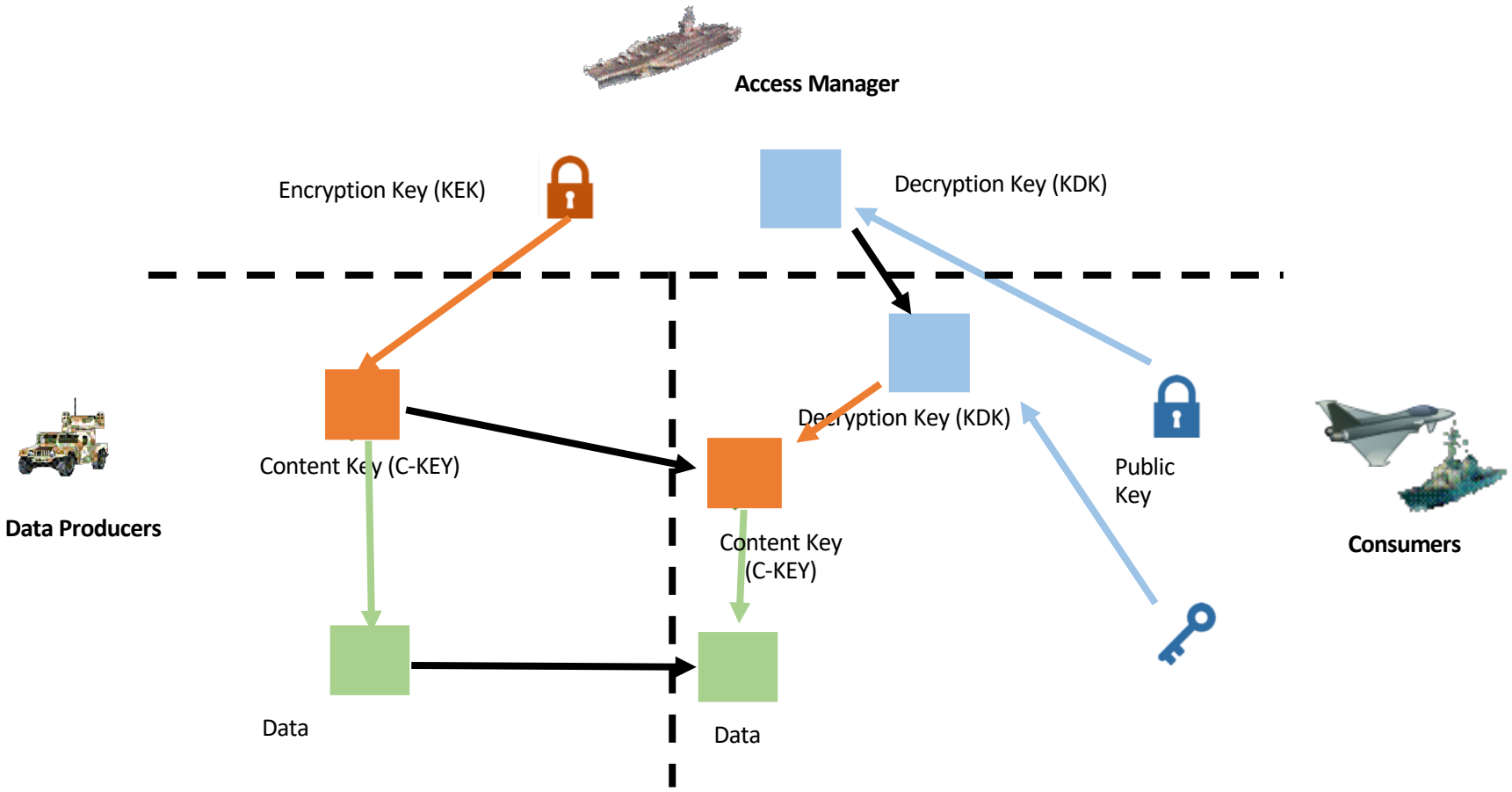


Name-Based Access Control (NAC)

- Access Controller – Base
  - Creates a list of encryption/decryption key pairs
    - encryption policy
  - Control whom to distribute the corresponding decryption keys
    - access policy
- Producers (Encryptors) – HMMV
  - Fetch the right encryption keys to encrypt data
- Consumers (Decryptor)
  - Fetch the right decryption keys to decrypt data



NAC Process


Access Manager
Encryption Key (KEK)
Decryption Key (KDK)
Content Key (C-KEY)
Decryption Key (KDK)
Public Key
Data Producers
Content Key (C-KEY)
Consumers
Data
Data

Naming Policies

Access Manager

Encryption Policies

Key Encryption Key (KEK): per file transfer group

/Apps/Mike/NAC/**Files/Ships**/**KEK**/<version>
/Apps/Mike/NAC/**Files/Troops**/**KEK**/<version>
/Apps/Mike/NAC/**Files/Xyz**/**KEK**/<version>
…

Access Policies

Per authorized participants Key Decryption Key (KDK)

/Apps/Mike/NAC/**Files/Ships**/**KDK**/<version>/ENCRYPTED-BY/…
…
/Apps/Mike/NAC/**Files/Troops**/**KDK**/<version>/ENCRYPTED-BY/…
…
/Apps/Mike/NAC/**Files/Xyz**/**KDK**/<version>/ENCRYPTED-BY/…
…

Naming Encryption Keys

Data Producers

From Access Manager / provisioned or dedicated data owner storage

- Fetches and stores KEK for the configured with access prefix

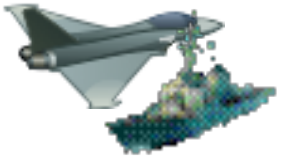
Interest ->
/Apps/Mike/NAC/**Files/Xyz**/**KEK**/<version>

- Encrypts input data using CK, returns encrypted content
- Exact name of the corresponding CK data is embedded in the encrypted content

- Generates (re-generates) symmetric Content Key (CK)
- Publishes CK data under configured namespace, encrypted by KEK

Data:
/Apps/Files/AreaA/SoldierA/timestamp/segment/…
/ENCRYPTED-BY
/Apps/Mike/NAC/**Files/Xyz**/**KEK**/<version>

Access to Protected Data



Data Consumers

- Fetch the encrypted Content Data
- Get the name of the corresponding CK: CK name is embedded in the encrypted content

From Encryptor / from same place as data

- Fetches CK data for the name extracted from input encrypted payload

Interest->
/Apps/Files/AreaA/SoldierA/CK/<key-id>

- Fetches KDK, name extracted from CK name + own configured access key name

Interest->
/Apps/Mike/NAC**/Files/Xyz/KDK/**<version>
/ENCRYPTED-BY
/Apps/Files/SoldierA**/Key/<key-id>**

From Access Manager / provisioned or dedicated data owner storage

name of user's key (e.g., same as for data authentication)

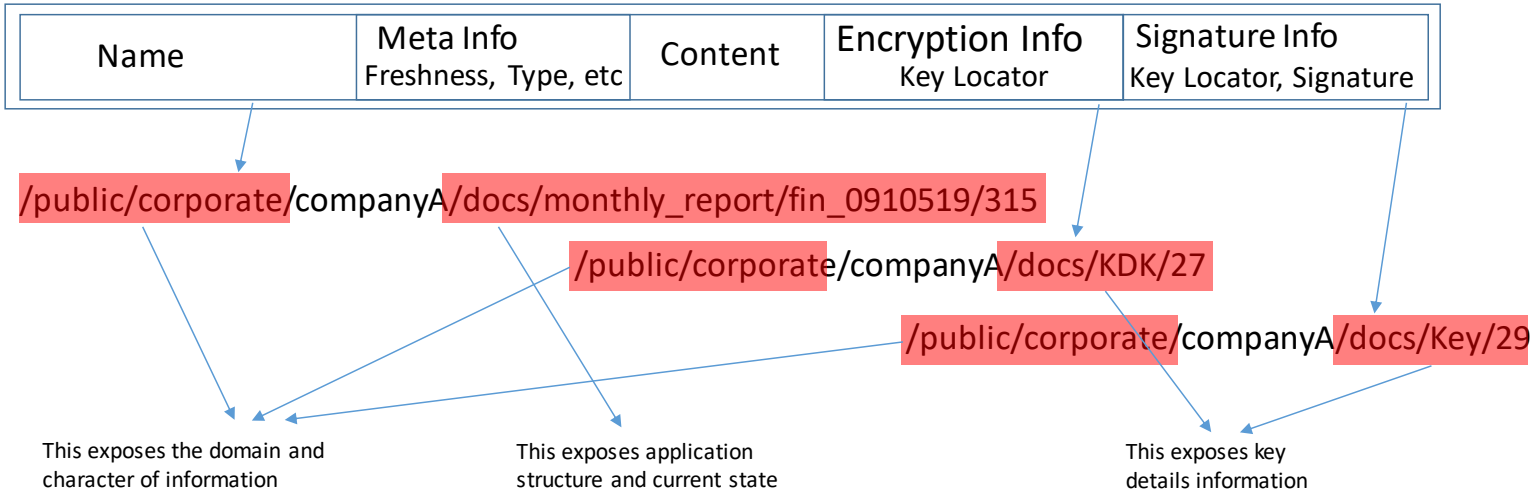
Name Privacy

Name Privacy

- Privacy of names extends beyond the application current state:
  - It exposes the state machine
    - retrieving certain data causes retrieval of a different data
  - It exposes dependencies between functionalities: sync vs. data plane
  - It exposes various time parameters of the app
- (Lack of) privacy for names brings up other security concerns:
  - In network targeted attacks against (particular) applications and functionalities
  - Targeted (D)DoS attacks via spurious Interests
    - Producer Denial of Service, PIT exhaustion
  - Content Store exhaustion
  - Possible modification of forwarding plane via spurious Interests

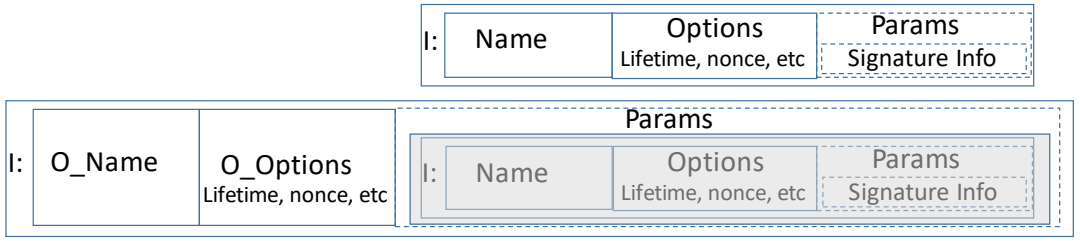
Name Privacy Considerations

- Secure Encapsulation
  - General requirements
  - Common approaches and tradeoffs
  - Approach details
  - Other Considerations

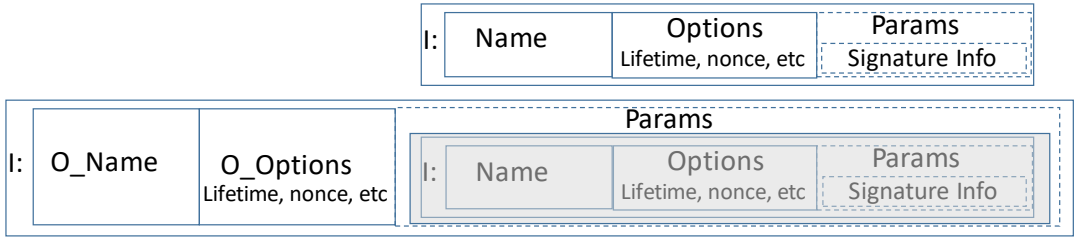


Encapsulation Model

- Outer Interest carries
  - encrypted inner interest as
    - new name component
    - new name
    - interest parameters
  - information to decrypt



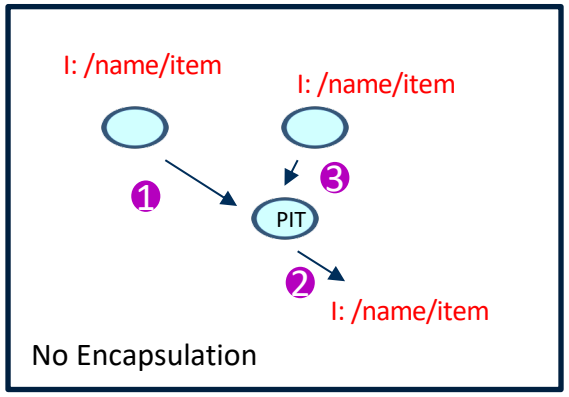
- Outer Data carries
  - encrypted data as content
  - information to decrypt

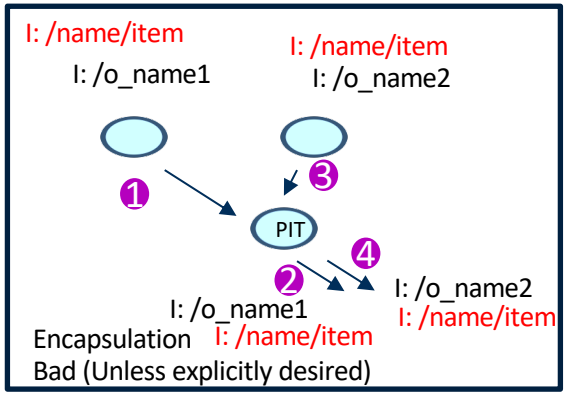


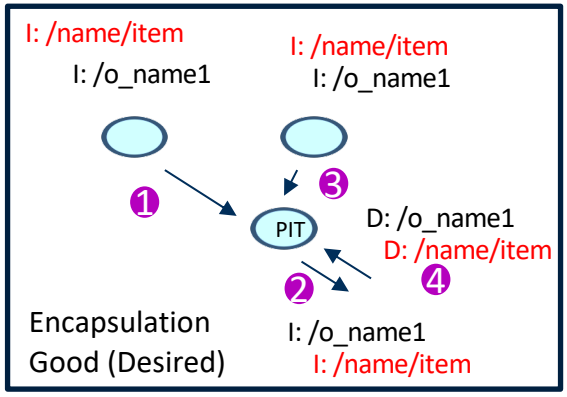
Name Encryption General Requirements

1. Must support Interest aggregation
2. Must support in-network data caching
3. Support real-time data fetching
   - I.e. Must-be-Fresh flag in Interest, FreshnessPeriod in Data packet
4. In-network name discovery/routing
   - I.e. fetching by name prefix, instead of a full data packet name

Implies one-to-one mapping between inner name and outer name

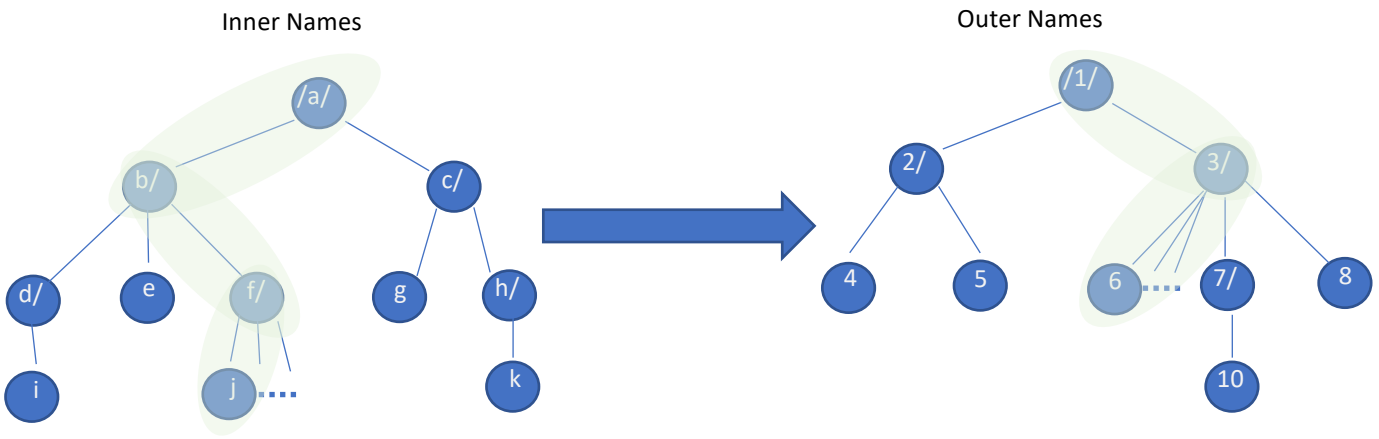






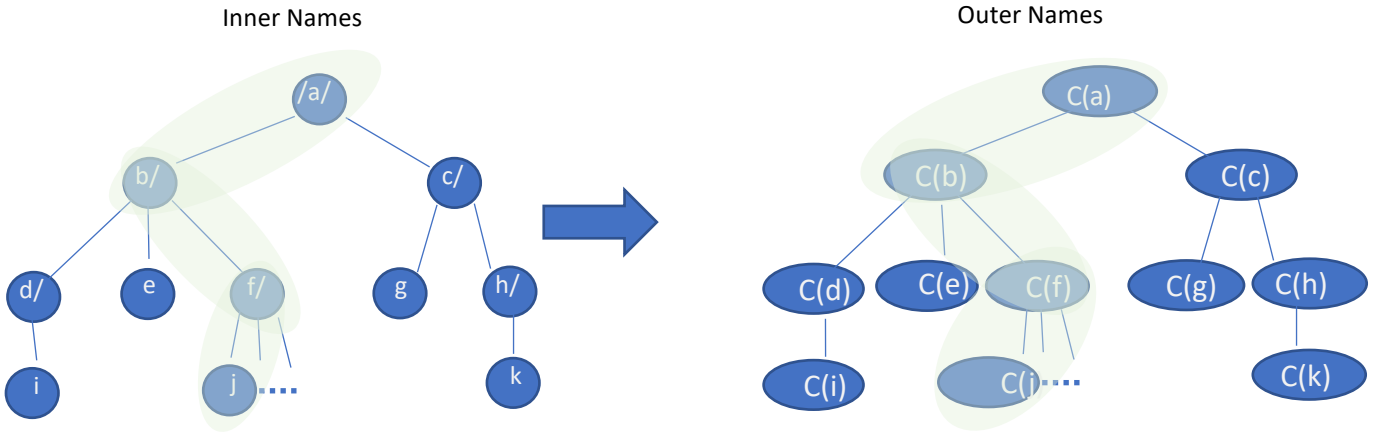
Encryption Function For NDN Names: A Tree View

milcom
Military Communications for the 21st Century
November 12-14, 2019 • Norfolk, VA, USA
Defining Multi-Domain Command and Control



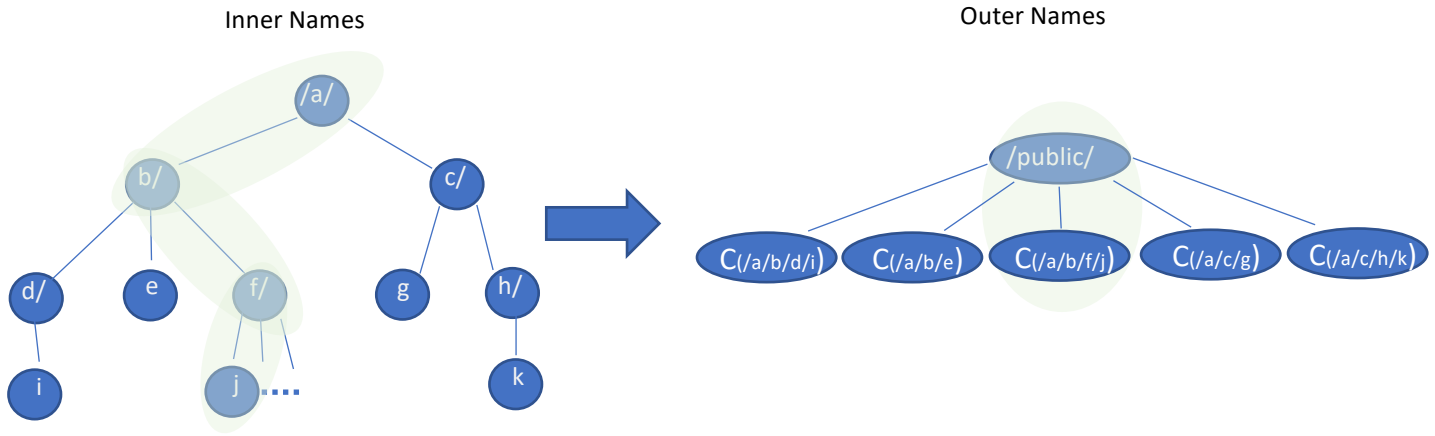
- NDN Namespaces can be represented as trees where nodes hold name components
- A Name is represented by a path between the root node and a leaf node
  - Sometimes to an intermediary node (see special case later)
- Encapsulation entails defining a mapping function between two such trees
- Encryption entails defining such a mapping that additionally preserves the confidentiality of the inner names
  - Cryptographic one way transformation

Encryption Function For NDN Names: Tree Structure Preserving



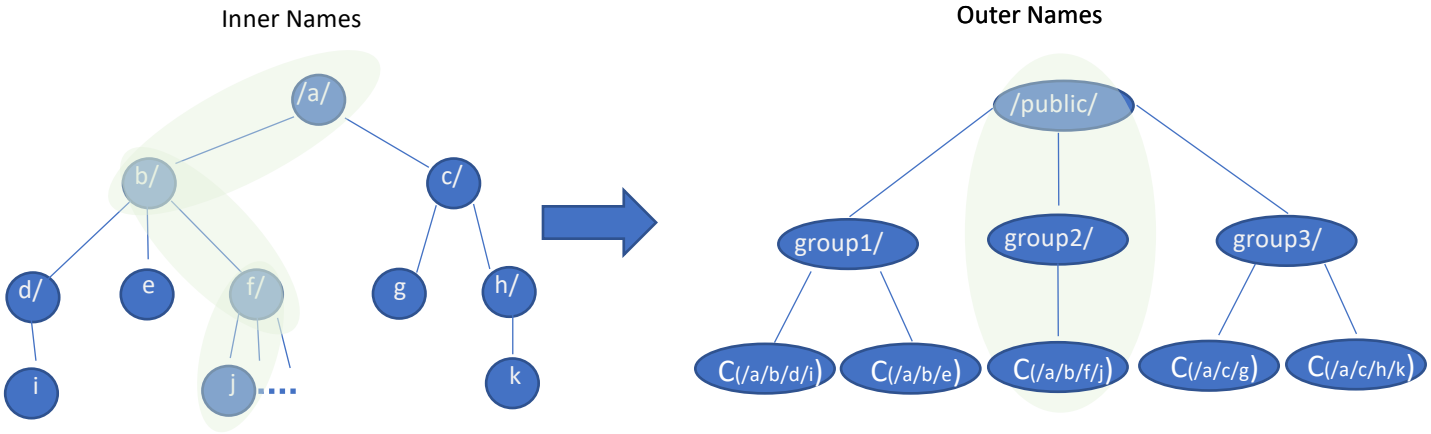
- C(name) represents a cypher-text of a name component
  - For performance reasons a cryptographic hash function might be used instead
  - Instead a name component, a partial path can be used instead
- Adversary will be able to reconstruct the structure of the inner name
  - Will be able to learn (quickly) the state of the application
- Network semi-friendly
- No issues with prefixing, see later

Encryption Function For NDN Names: Tree Flattening



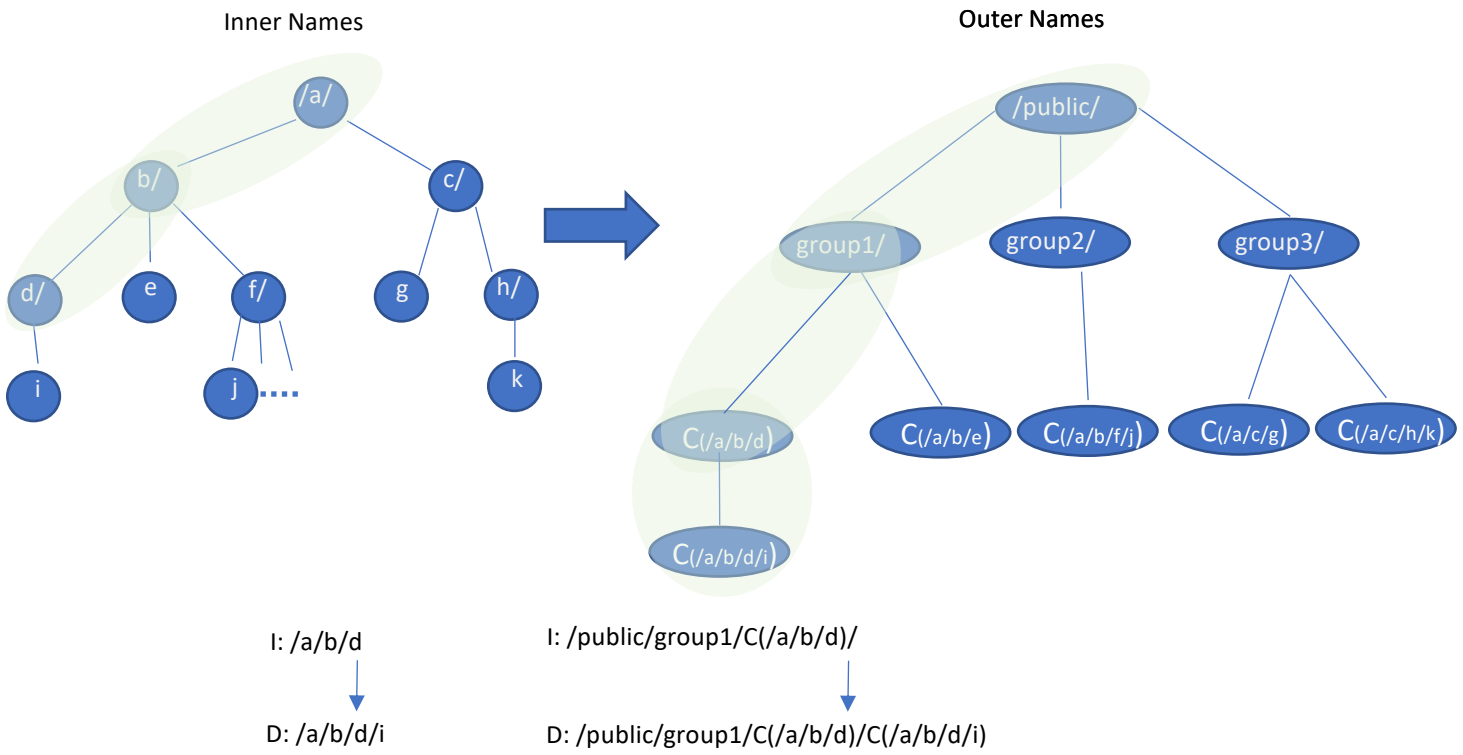
- C(name) represents a cypher-text of a name
  - For performance reasons a cryptographic hash function might be used instead
- Adversary cannot reconstruct structure of the inner name
- Very network unfriendly
- Issues with prefixing

Encryption Function For NDN Names: Network Friendly



- C(name) represents a cypher-text of a name
  - For performance reasons a cryptographic hash function might be used instead
- Hybrid solution
  - Provide partial mapping between inner names and outer prefix
    - Outer prefix may have variable depth/length
  - Outer leaf computed based on the cypher-text of the inner name
- Mapping performed based on name schema, according to name matching rules
  - Schema may support both previous alternatives
- Mapping between inner name and outer prefix uses flexible mapping that balances network friendliness with confidentiality

Encryption Function For NDN Names: Prefixing



- Supports prefixing for /a/b/d

Conclusion

- Data-centric security
- Leverage naming of everything
  - trust schema to authorize access
  - signing key management
  - encryption key management
- Name (interest) privacy
  - configurable trade offs