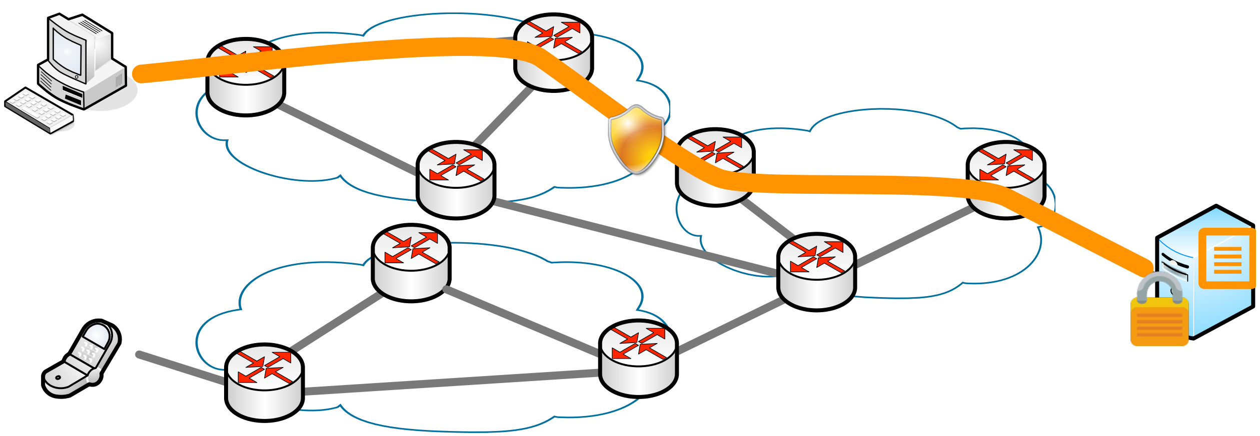
Usable Security of Named Data Networking

Yingdi Yu

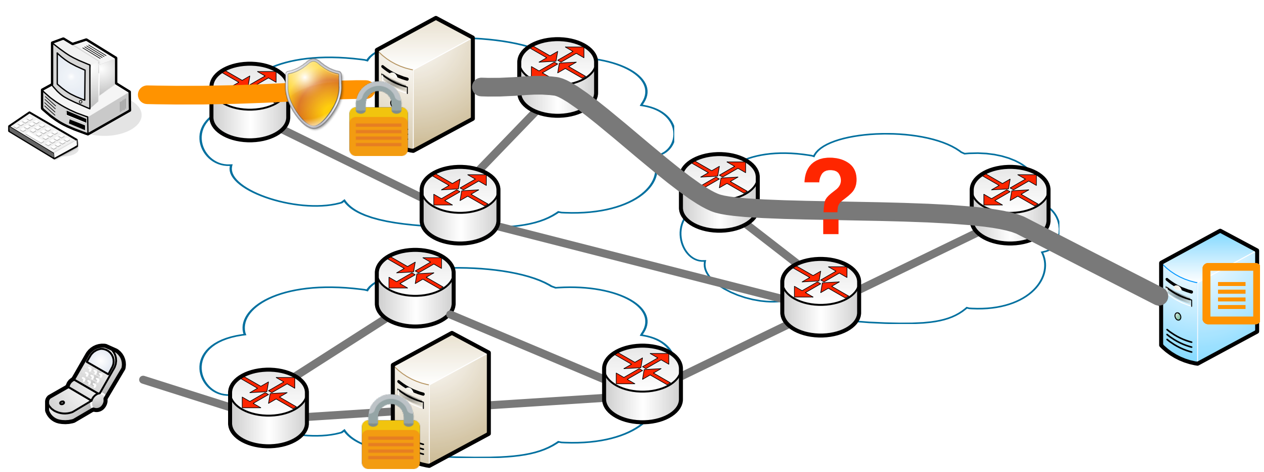
Traditional communication model of Internet

- Speaking to a host
  - end-to-end channel
- Communication security
  - container-based authenticity: X.509, Certificate Authority
  - channel-based confidentiality: IPSec, TLS/SSL



New communication vs. Old security

- Content Distribution Network (CDN)
  - multiple containers to secure
  - no end-to-end channel

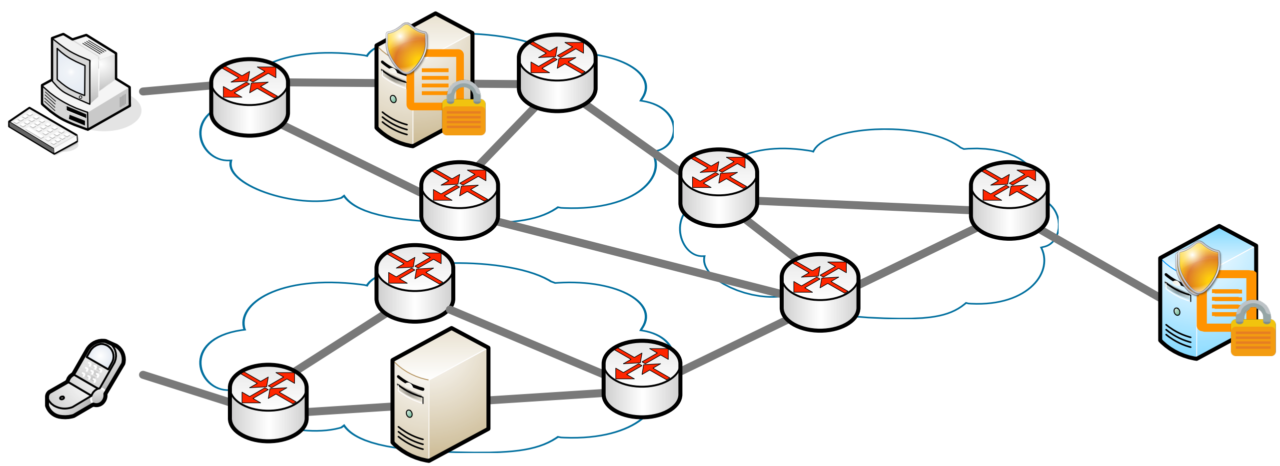


New communication vs. Old security

- Delay Tolerant Network (DTN)
    - temporary data container
    - no instantaneous end-to-end channel

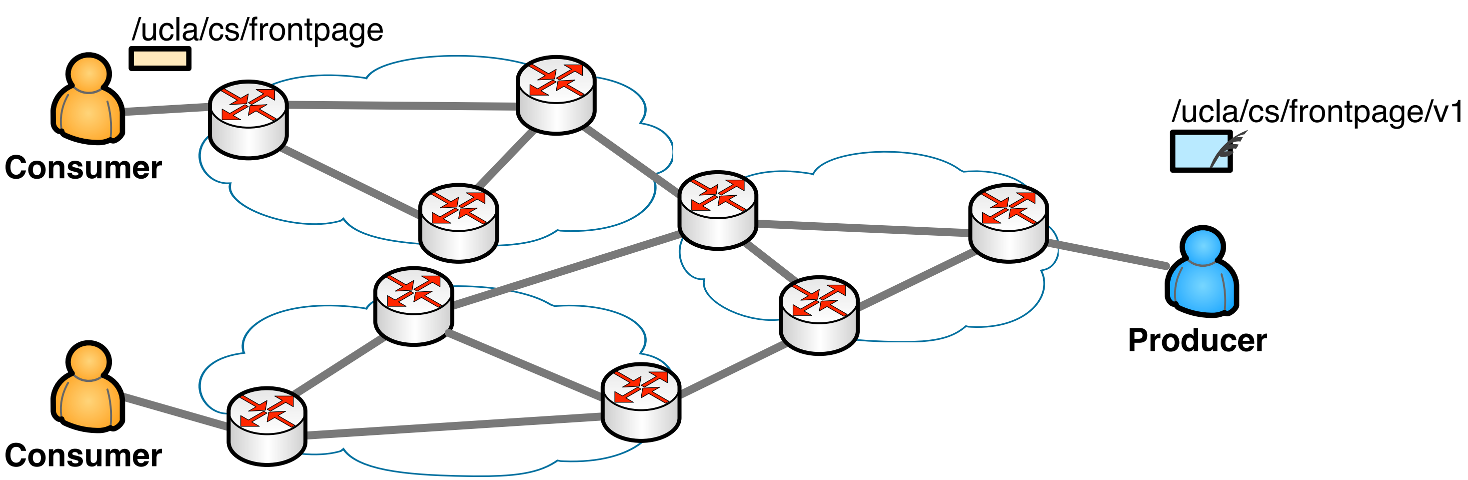
New security model is desired!

- No trustworthy container, no end-to-end encrypted channel
- Data-centric security: let's secure **data** directly!
  - authenticate data rather than container
  - encrypt data instead of channel



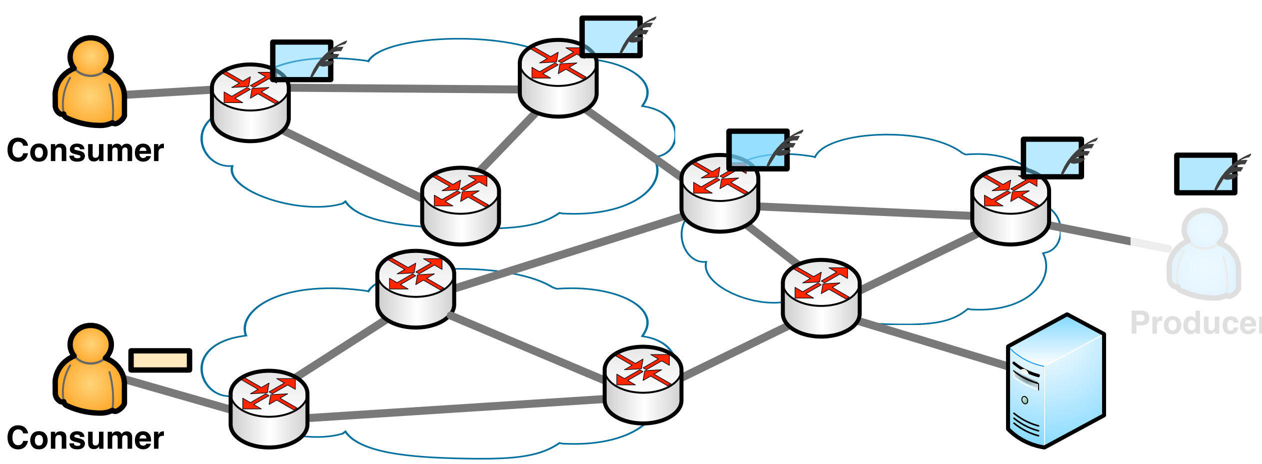
Named Data Networking

- Data-centric communication primitives
  - retrieve data by name rather by container address
  - **Interest Packet**: expressed by consumer, forwarded according to name
  - **Data Packet**: made by producer, forwarded along reverse path



Efficient & flexible data delivery

- Data can be picked anywhere
  - in-network caching
- Does not require instantaneous communication
  - producer can go offline
  - store pre-created data in third party storage



Built-in data authenticity

- Per packet signature
  - privilege separation: different data signed by different keys
- Retrieve public key as data
  - same authentication procedure
- Data carrying public key is a certificate
  - more powerful

NDN Data Packet

Name: /ucla/cs/alice/thesis/v_3/s_8

Content: ...

SignatureInfo:
KeyLocator: /ucla/cs/alice/KEY/2
ValidityPeriod: [2015/5/2, 2016/5/2)
...

Signature Bits: ...

NDN Certificate

Name: /ucla/cs/alice/KEY/2

Content:
6d:32:8d:23:a9:b0:89:...

SignatureInfo:
SignatureType: RSA-SHA256
KeyLocator: /ucla/cs/KEY/7
ValidityPeriod: [2015/1/1, 2017/1/1)
...

Signature Bits:
cd:ca:70:72:7b:ff:a8:...

X 509 Certificate

Subject Name

Subject Public Key Info

Certificate Signature Algorithm

Issuer Name

Validity Period

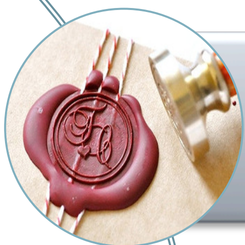
Certificate Signature

But how to utilize those features?

- Developers turn off security as the first step
  - fake signature
  - skip authentication
  - wish no one is eavesdropping

- Can we make security easier for developers?
  - automate data authentication
  - automate data encryption
  - minimize maintenance overhead

Outline

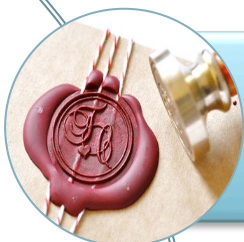


Automating Data-Centric Authenticity

Authenticating Long-Lived Data

Automating Data-Centric Confidentiality

Outline



Automating Data-Centric Authenticity

Authenticating Long-Lived Data

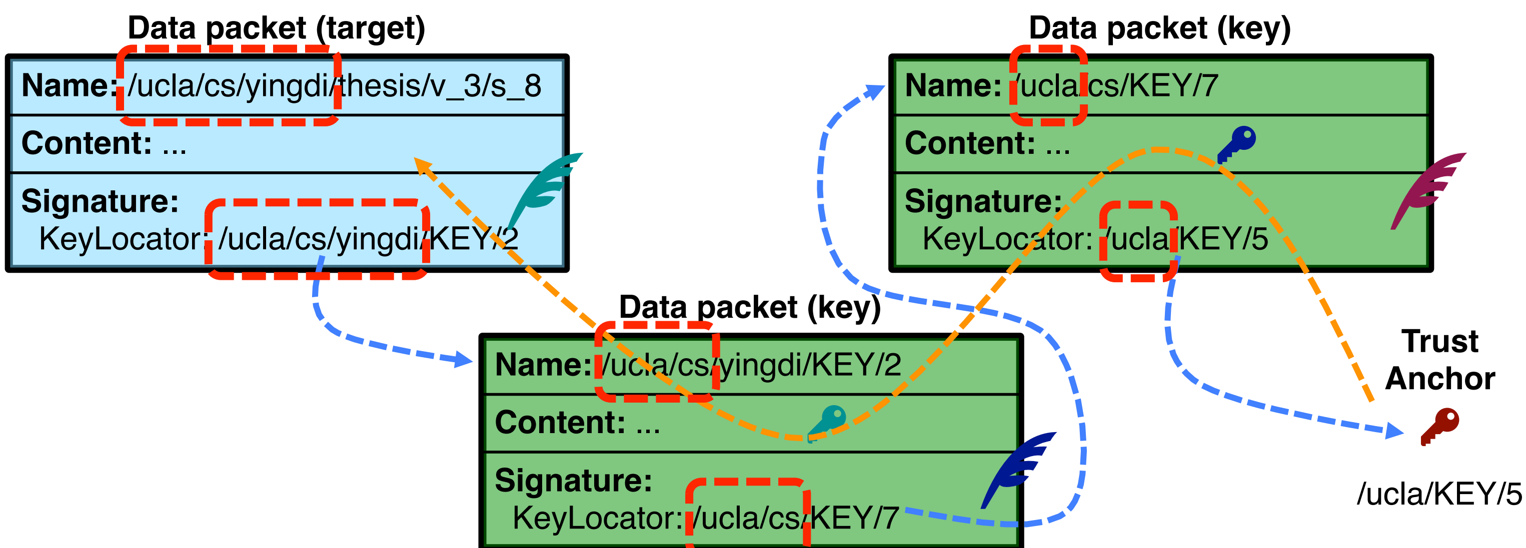
Automating Data-Centric Confidentiality

Trust chain

- Recursively retrieve key until reach a **trust anchor**
  - a pre-trusted key

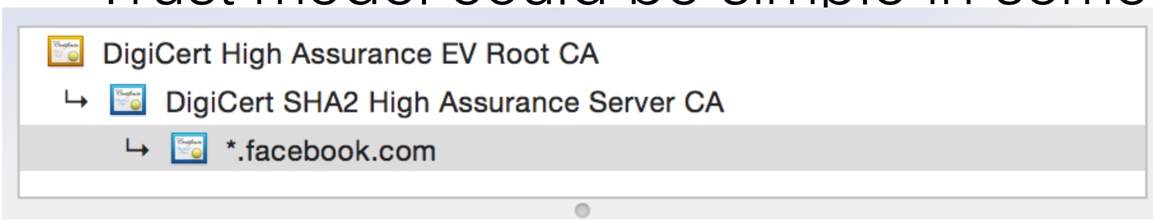
Trust Model

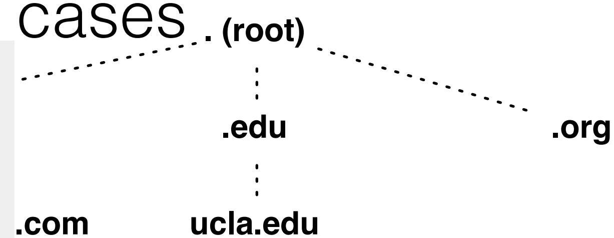
- Constrained by **trust derivation rules**
  - is data (or key) signed by a trusted producer (or issuer) ?
- Validate signature



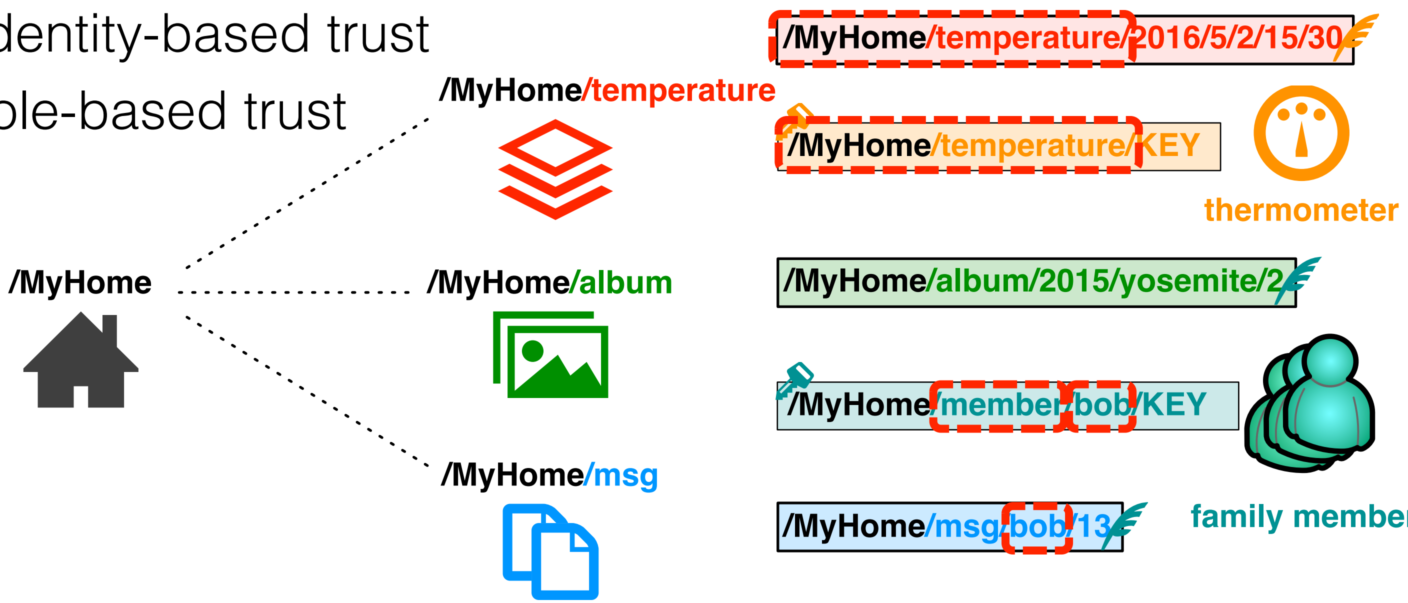
Diversity of trust models

- Trust model could be simple in some cases





- Application specific in general
  - capability-based trust
  - identity-based trust
  - role-based trust



NDN insight

- Name is a general expression
  - can refer to identity, capability, role, …
- Any trust model can be expressed as a list of relationship between data name and key name

Schematize the trust
Automate data authentication

- Data auth … easily if we have
  - a name-based policy language to express trust model
  - a library to perform authentication according to the policy

Describe trust relationship in name

- Relationship between data and key names

/My/home/msg/bob/13 ← /My/home/member/bob/KEY
/My/home/msg/alice/15 ← /My/home/member/alice/KEY

- Generalized as name pattern

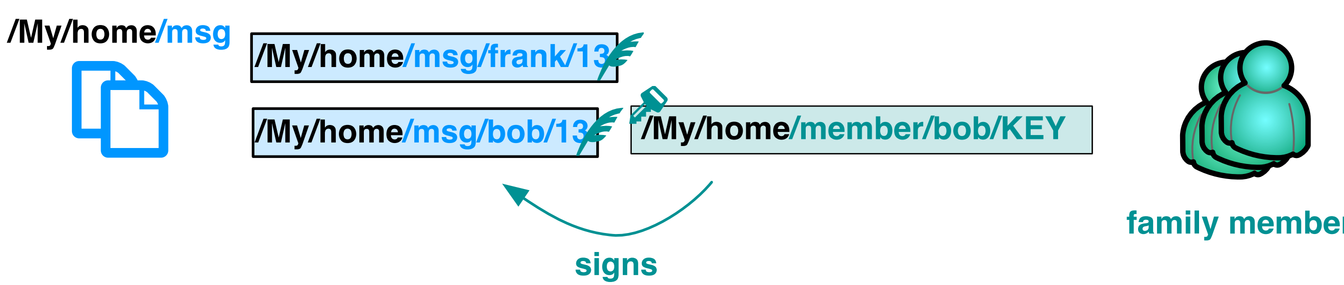
home_prefix + “home” + “msg” + user + msg_id
home_prefix + “home” + “member” + user + “KEY”

signs

- Regex-based syntax

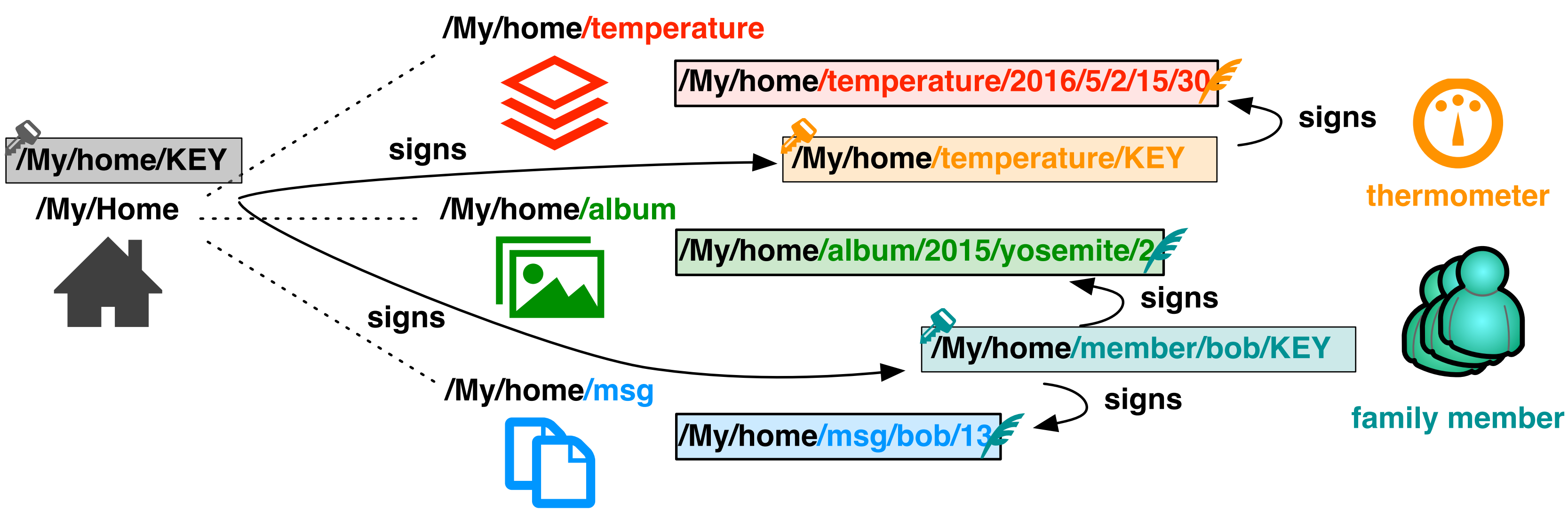
(<>*)<home><msg>([user_id])<>
\1<home><member>\2<KEY>

signs



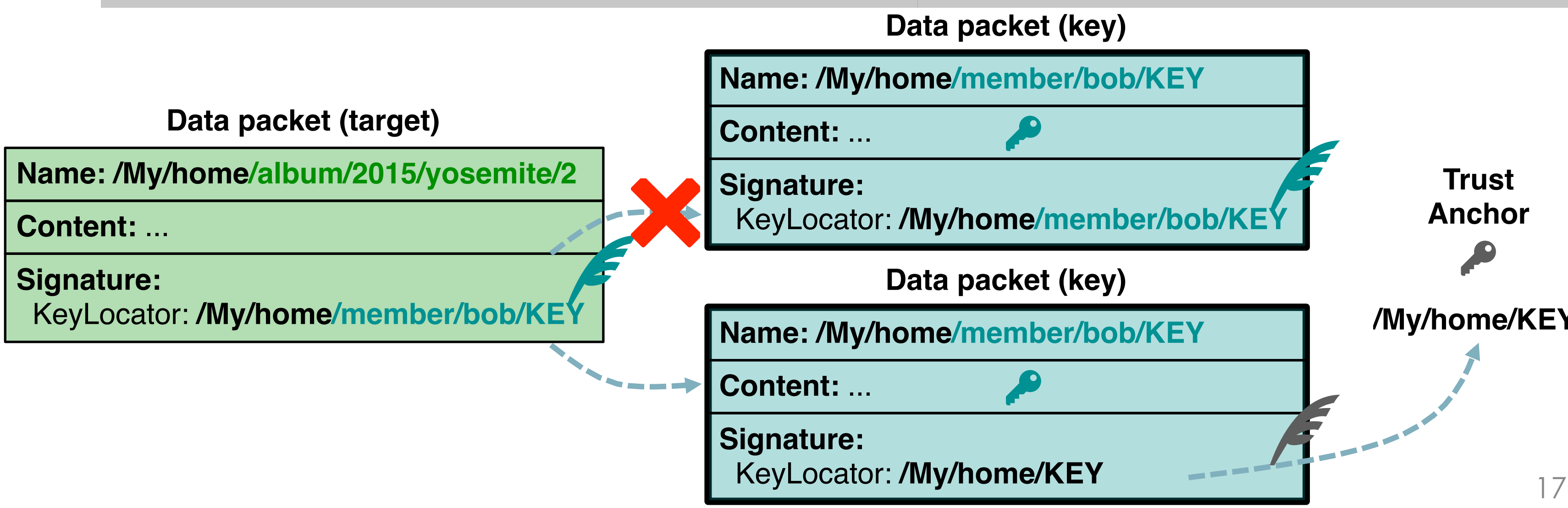
Trust schema

Rule ID	Data Name	Key Name
msg	(<>*)<home><msg>**([user])**<>	\1<home><member>\2<KEY>
album	(<>*)<home><album><><><>	\1<home><member>[user]<KEY>
temp	(<>*)<home><temperature><><><><><>	\1<home><temperature><KEY>
member	(<>*)<home><member>**([user])**<KEY>	\1<home><KEY>
therm	(<>*)<home><temperature><KEY>	\1<home><KEY>
root	(<>*)<home><KEY>	/My/home/KEY 30:b4:82:9c:45:…



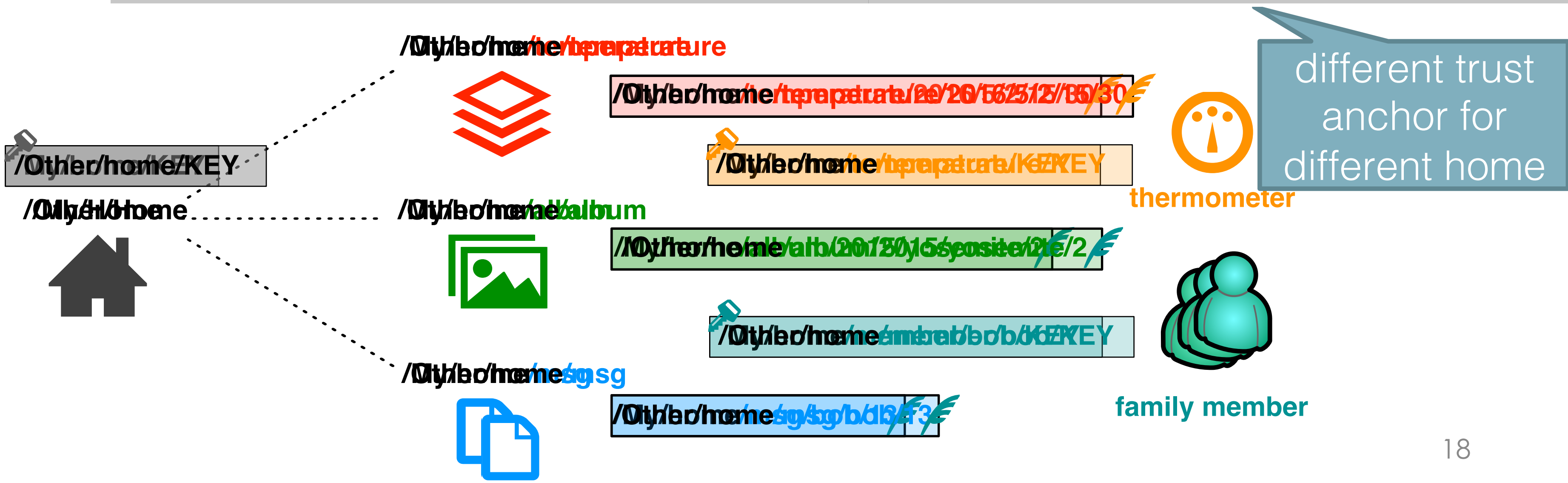
Trust chain construction

Rule ID	Data Name	Key Name
msg	(<>*)<home><msg>([user])<>	\1<home><member>\2<KEY>
album	(<>*)<home><album><><><>	\1<home><member>[user]<KEY>
temp	(<>*)<home><temperature><><><><><>	\1<home><temperature><KEY>
member	(<>*)<home><member>([user])<KEY>	\1<home><KEY>
therm	(<>*)<home><temperature><KEY>	\1<home><KEY>
root	(<>*)<home><KEY>	/My/home/KEY 30:b4:82:9c:45:…



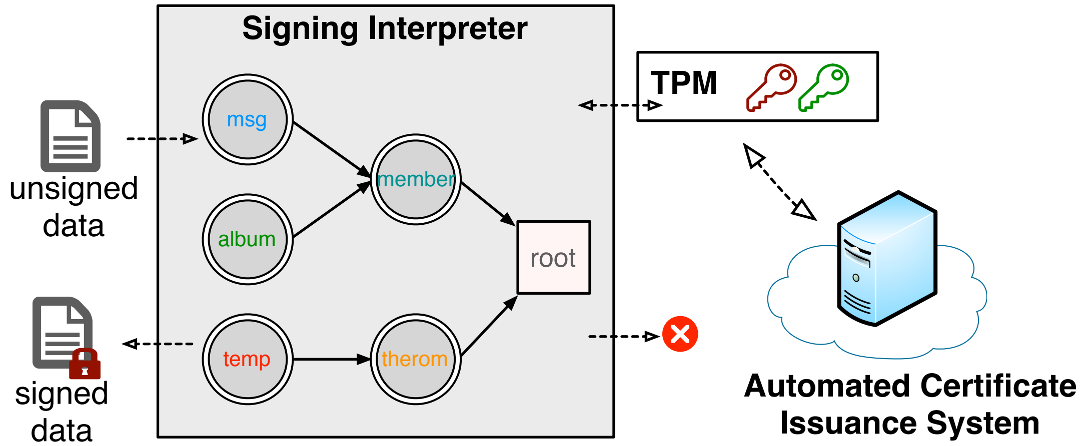
Re-usability

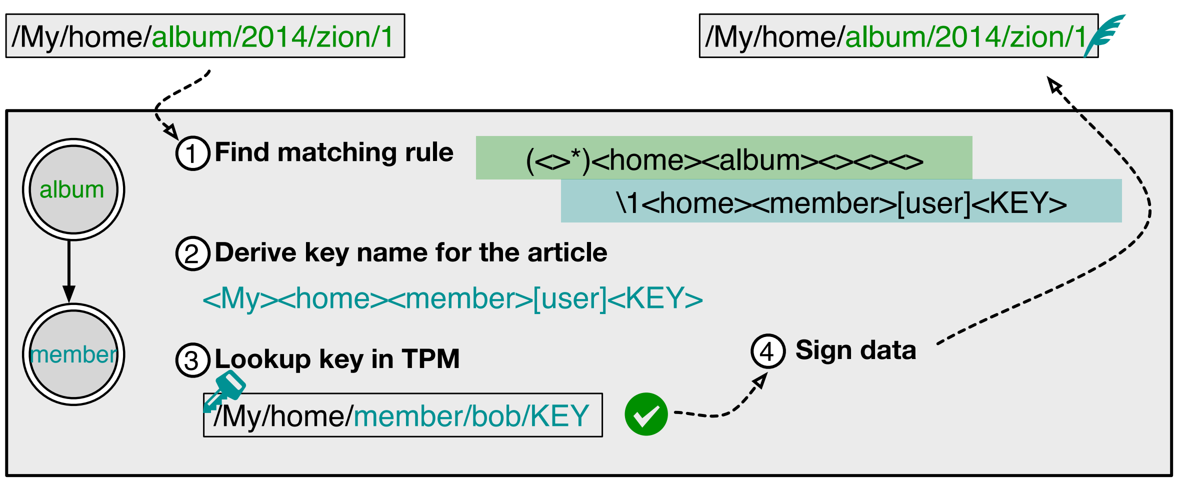
Rule ID	Data Name	Key Name
msg	(<>*)<home><msg>([user])<>	\1<home><member>\2<KEY>
album	(<>*)<home><album><><><>	\1<home><member>[user]<KEY>
temp	(<>*)<home><temperature><><><><><>	\1<home><temperature><KEY>
member	(<>*)<home><member>([user])<KEY>	\1<home><KEY>
therm	(<>*)<home><temperature><KEY>	\1<home><KEY>
root	(<>*)<home><KEY>	/Other/home/KEY 9c:45:30:b4:82:…



Automated Signing

- Signing Interpreter
- Determine signing key
- Request certificate if needed





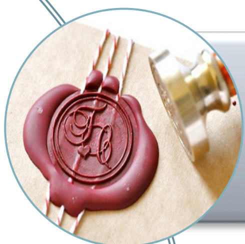
Implementation

- Available in all the NDN platform libraries
  - ndn-cxx: http://www.github.com/named-data/ndn-cxx
  - NDN-CCL: http://named-data.net/codebase/platform/ndn-ccl/
- Powers data and interest authentication in:
  - NFD: NDN Forwarding
  - NLSR: NDN Link State Routing Protocol
  - NDNS: NDN Domain Name System
  - Repo-ng: NDN Data Repository
  - ChronoChat: server-less multi-party chat application over NDN

Summary

- Trust schema is a **general expression** of NDN trust model
  - can be executed by any entity
- Trust schema is in **text format**
  - can be distributed as data packets
- A trust schema represents a **security design pattern**
  - regulate the behavior of applications
  - a set of common trust schemas

Outline



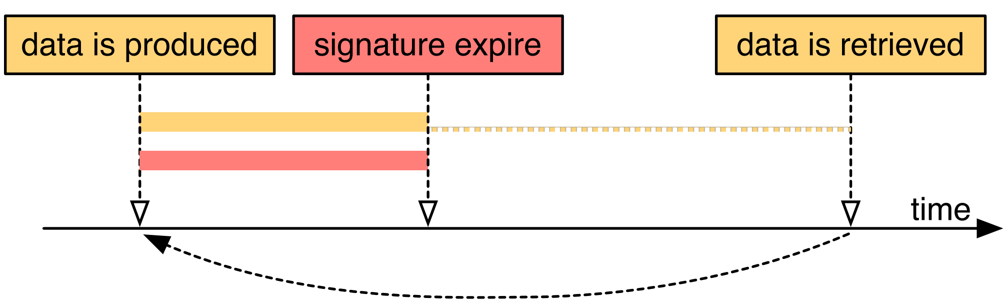
Automating Data-Centric Authenticity

Authenticating Long-Lived Data

Automating Data-Centric Confidentiality

Lifetime Mismatch

- Data lifetime is usually longer than its signature
  - crypto algorithm, key compromise, …
- Periodical re-signing is not the solution
  - will not scale in long term
  - data may outlive its producer
  - not a problem in channel based security
- After fact validation
  - verify signature validity at the time of production

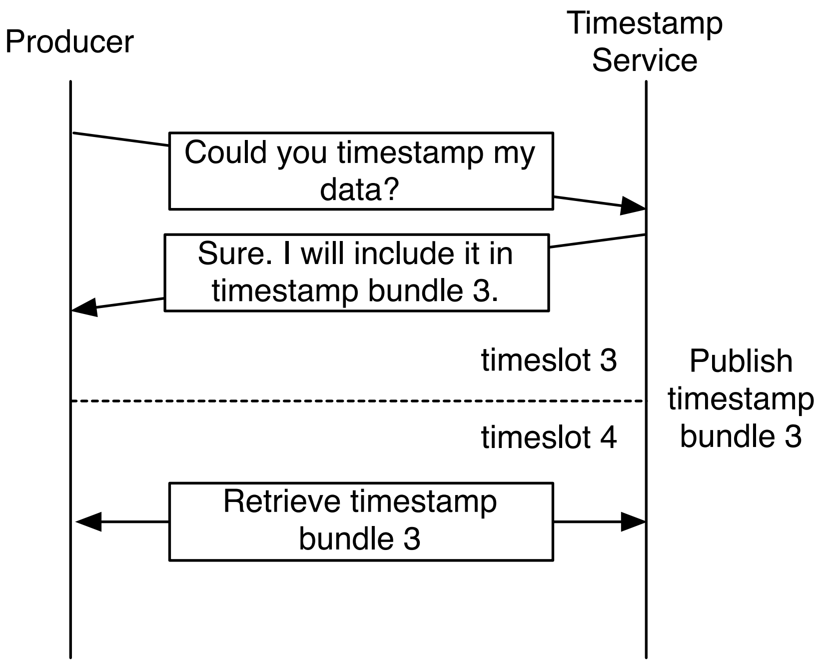


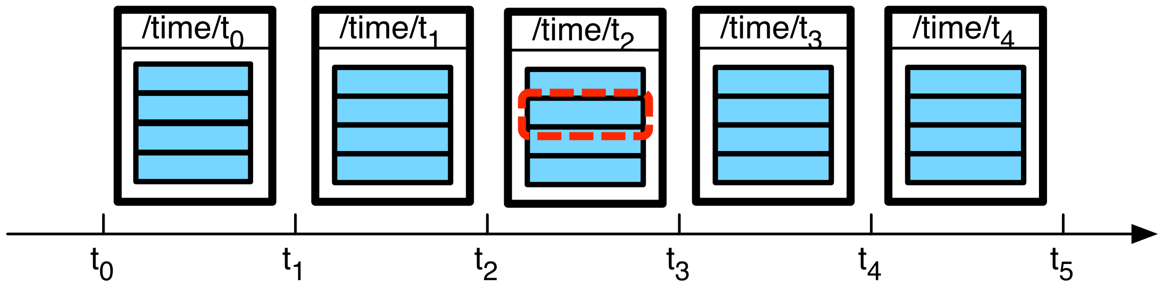
How to rollback the clock?

- Timestamp service
  - producer requests timestamp of data from the service
  - provide existence proof of data at a given time point
- Design challenges
  - how to tell the timestamp service is honest?
  - scale with the number of timestamp records

Verifiable timestamp

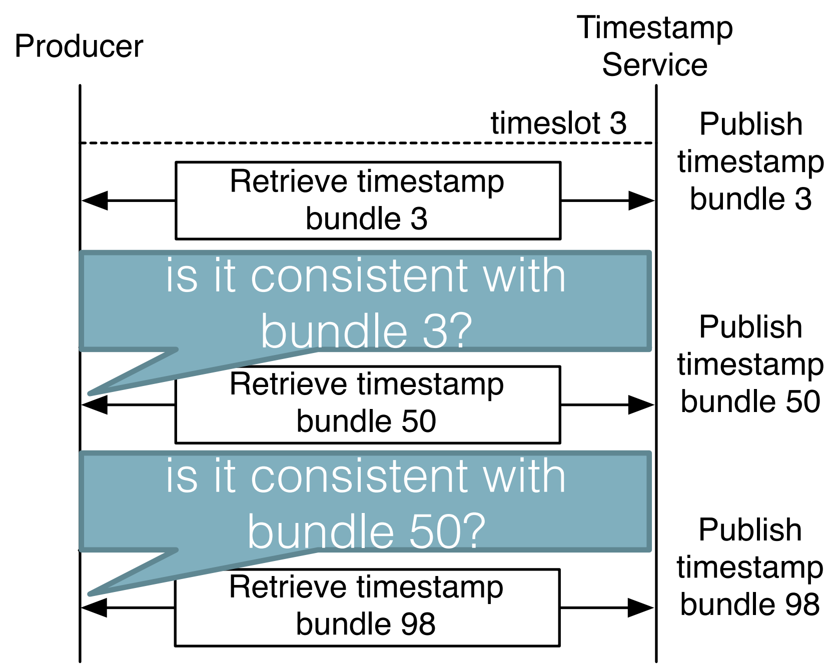
- Timestamp service periodically publishes a timestamp bundle
  - containing data received during the time period
- Producer requests including its data in a bundle
- Existence verification
  - check whether data is in the corresponding bundle

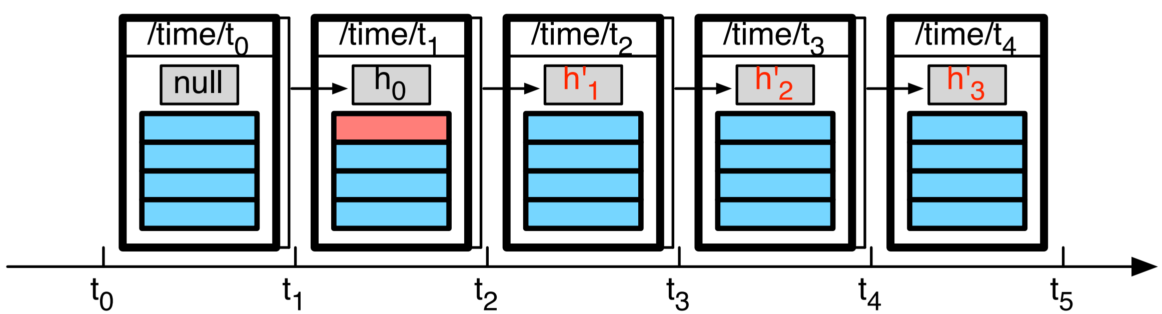




Consistent timestamp

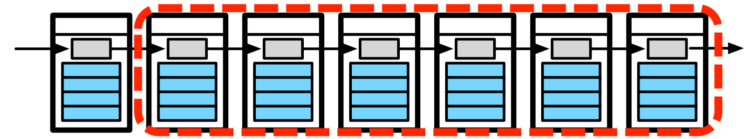
- Consistence verification
  - check whether the timestamp service modified the history
- A naïve solution: hash chain
  - each timestamp bundle fixes all the previous timestamp bundles
  - consumers and producers can verify consistency periodically





Reduce verification overhead

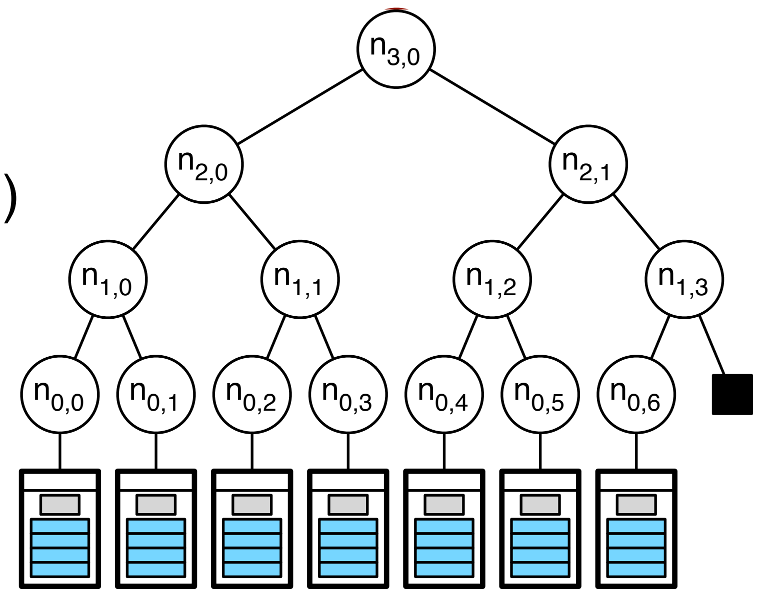
- Hash chain: O(m)
  - m: number of timeslots
  - 10-min timeslots for 20 years: 10^6



- k-ary Merkle tree:

  $$h_{i,n} = H(h_{i-1,nk^i} | h_{i-1,nk^i+1} | \ldots | h_{i-1,nk^i+k-1})$$

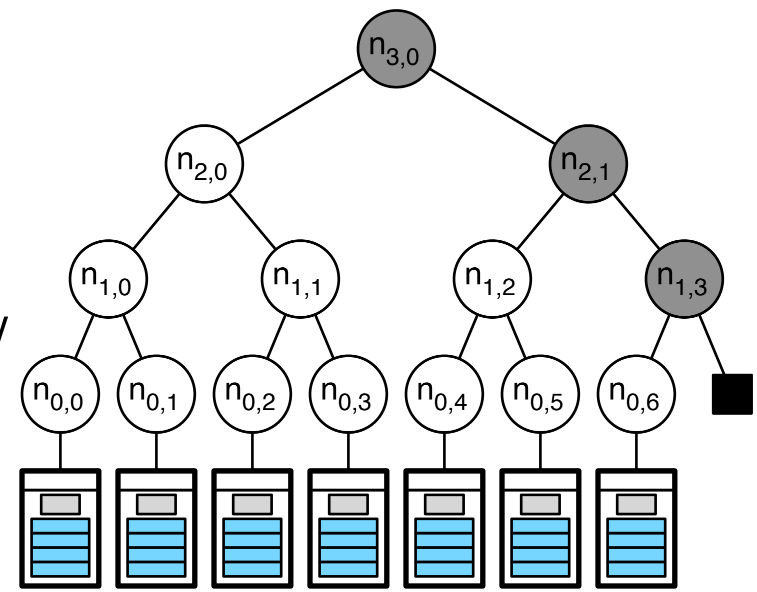
  - root hash as the state
  - existence verification:
    - $O(\log_k m)$
  - consistence verification:
    - $O(\log_k m)$
  - 20 years timestamps
    - 4 hash computations for 32-ary Merkle tree



Verification proof as data

- Proof is a pre-determined node set
  - simply publishes each node as data
  - consumer look up nodes necessary for verification

- Update nodes after adding a new timestamp bundle
  - complete nodes are not changed
  - at most one incomplete node at each layer



Node data

- Naming convention
  - uniquely identify a node in a particular state

/[tree_prefix]/[completeness]/[layer]/[index]/[hash]

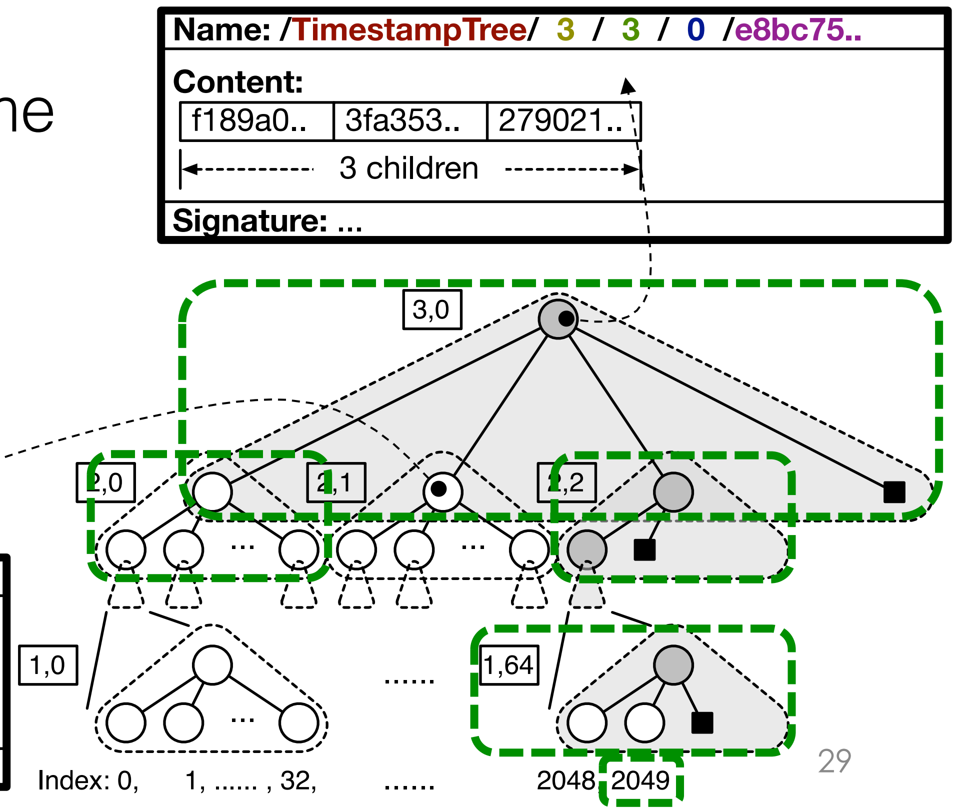
- Given a time point, the name of any node is determined

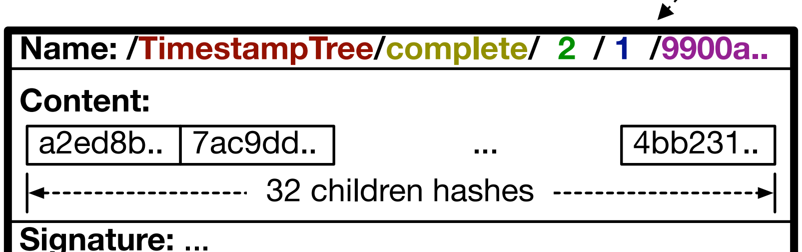
/TimestampTree/2050/1/64

/TimestampTree/complete/2/0

/TimestampTree/2050/2/2

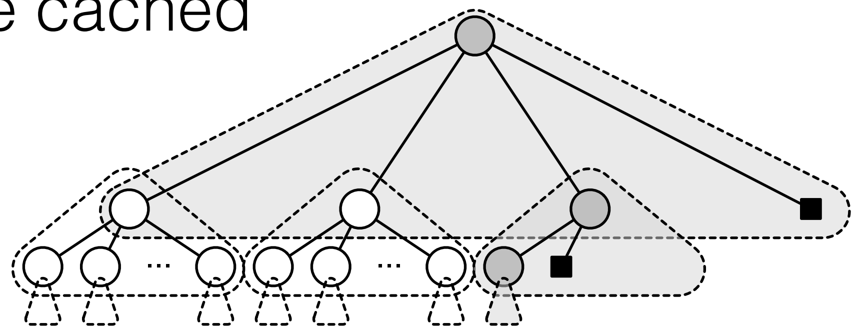
/TimestampTree/2050/3/0



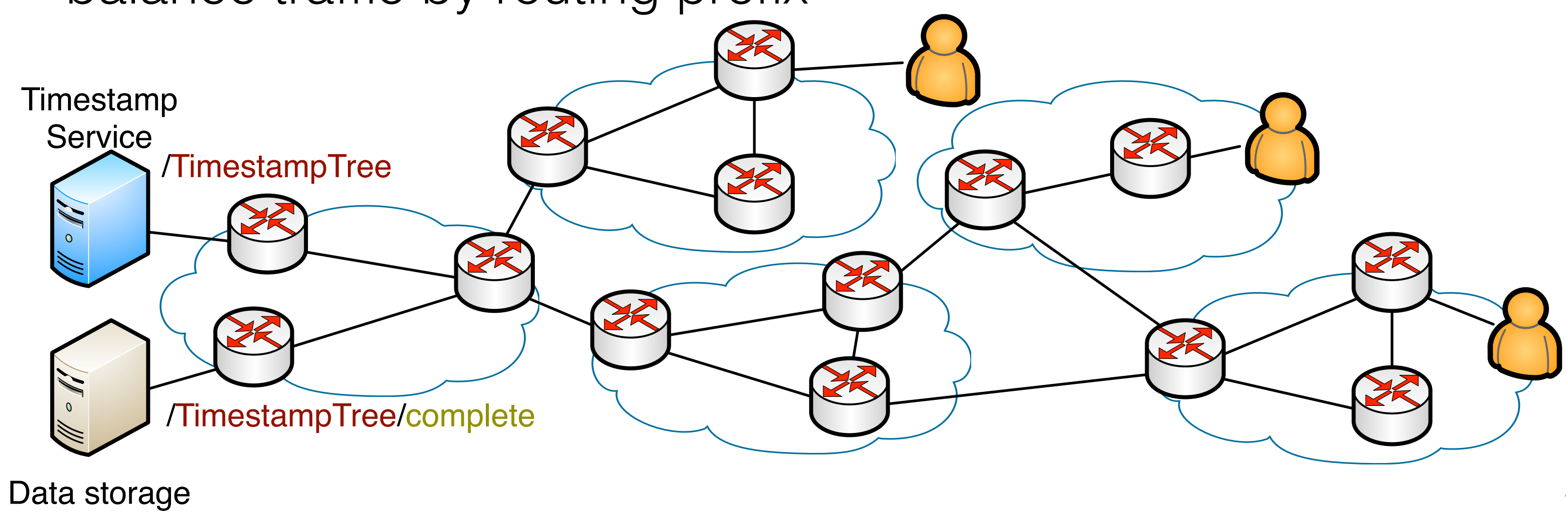


Node retrieval

- Nodes at higher layers are cached
  - more frequently retrieved
  - root node cached almost everywhere

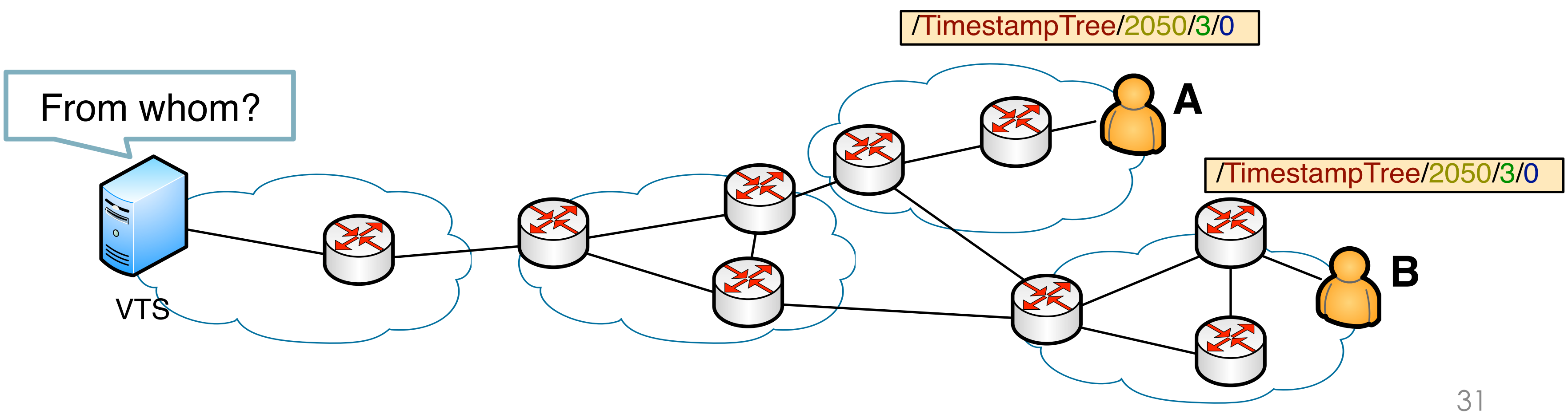


- Complete nodes can be served by dummy storage
  - balance traffic by routing prefix



Public auditing with Merkle tree

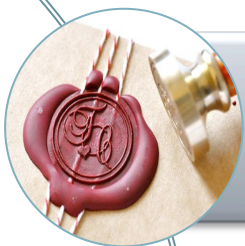
- All the users verify the consistence of timestamp service
  - occasionally retrieves the root
  - the more users, the more secure
    - single timestamp service for all the users
- Difficult to create double history
  - NDN interest does not carry sender address
  - Interest may not reach timestamp service (satisfied by cache)



Summary

- After fact validation is an authentication model for **non-instantaneous** communication
  - decouple the lifetime of data and signature
  - encourage the use of short-lived key
- **Untrustworthy** but **verifiable** timestamp service in NDN
  - borrow the concept public auditing concept from Certificate Transparency
  - publishing Merkle-tree as data simplifies verification query processing
  - absence of source address and efficient data distribution facilitates public auditing

Outline



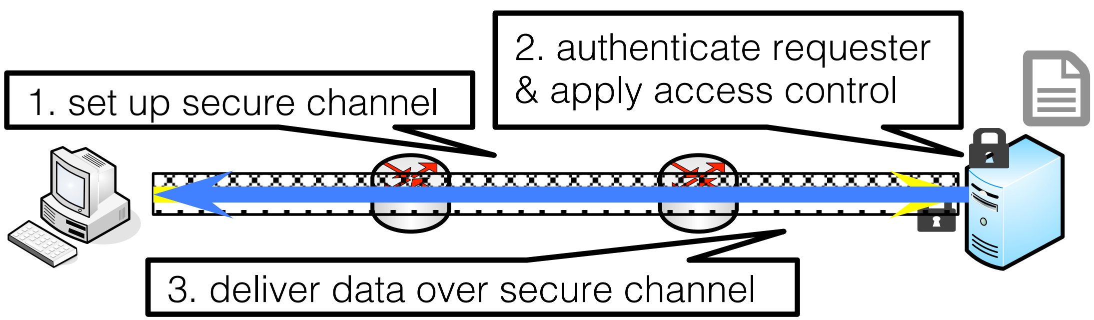
Automating Data-Centric Authenticity

Authenticating Long-Lived Data

Automating Data-Centric Confidentiality

Data confidentiality

- Current practice: perimeter-based security
  - data stays in trusted container
  - pass data to authorized users through an end-to-end secure channel
- Can we support data owner controlled confidentiality without trusted container and secured end-to-end channel?

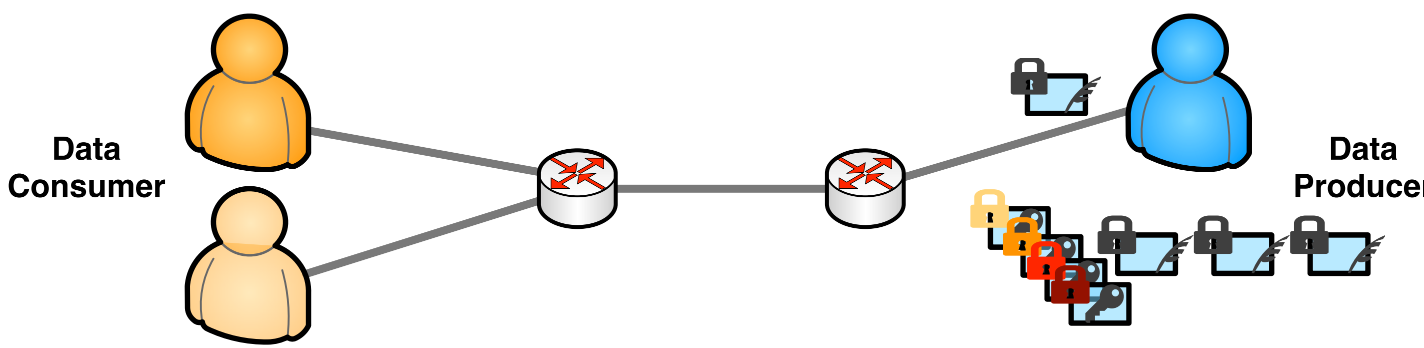


Data-centric confidentiality

- Encrypt data at the time of production
- Distribute decryption keys to authorized consumers
- Design challenges

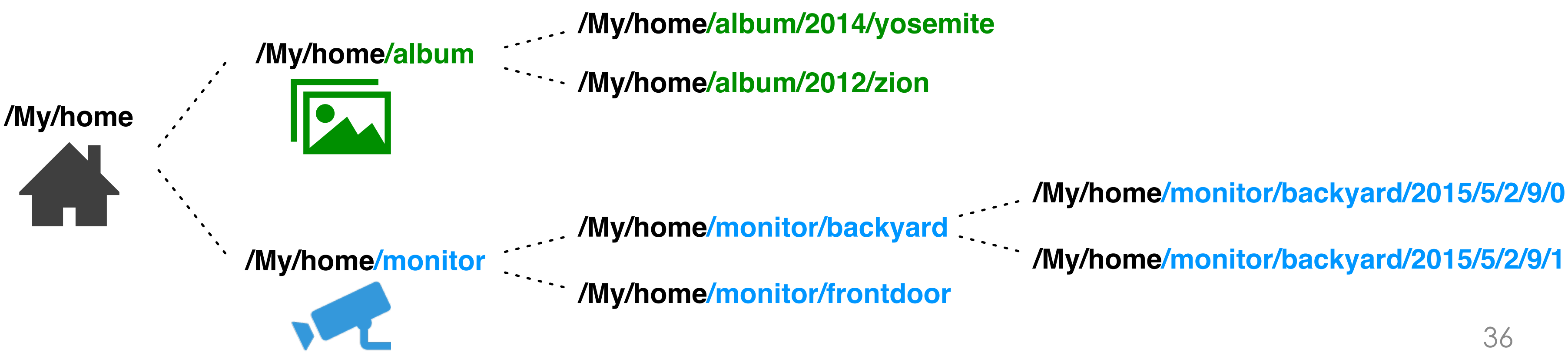
Specify privilege using hierarchical name
Publish encryption instruction as named keys

  - How to distribute decryption keys efficiently?



Distributed production & Dynamic sharing

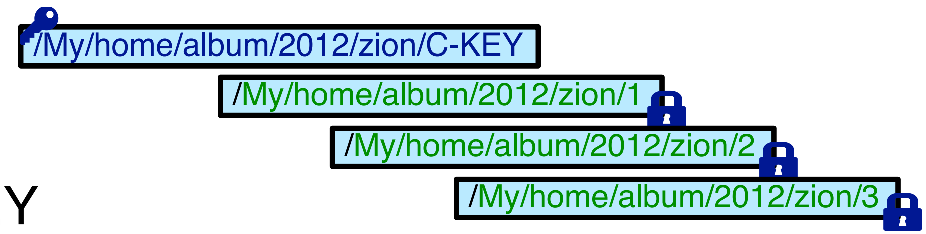
- Shared album in SmartHome
  - members produce photos at different sites in different years
  - shared with relatives later
    - no pre-knowledge about whom the photos will be shared with
- House surveillance video
  - produced by cameras in different rooms
  - allow security personel to watch the video when nobody at home
    - no pre-knowledge when family is out



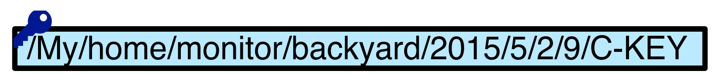
Content key

- Data is encrypted using a content key (C-KEY)
  - symmetric key
  - generated by producer

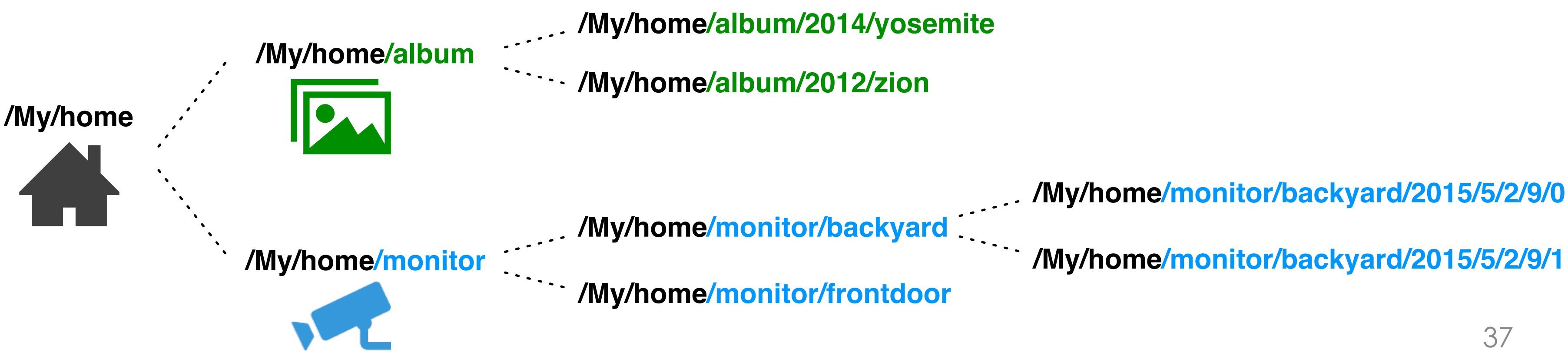
  /[content_namespace]/C-KEY



- Minimal access granularity
  - encrypt data under the namespace



- Distributed to authorized consumers eventually



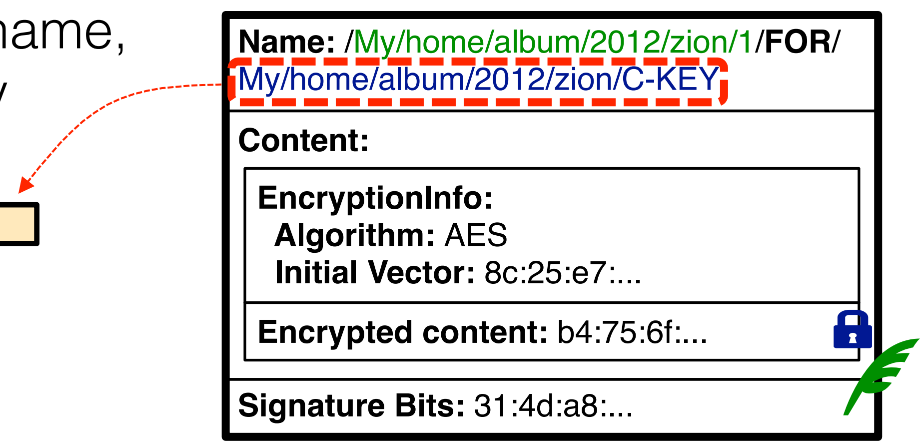
Encrypted data

- A data packet with encrypted content
  - encryption metadata
  - encrypted content

- Encryption key name is encoded in data name

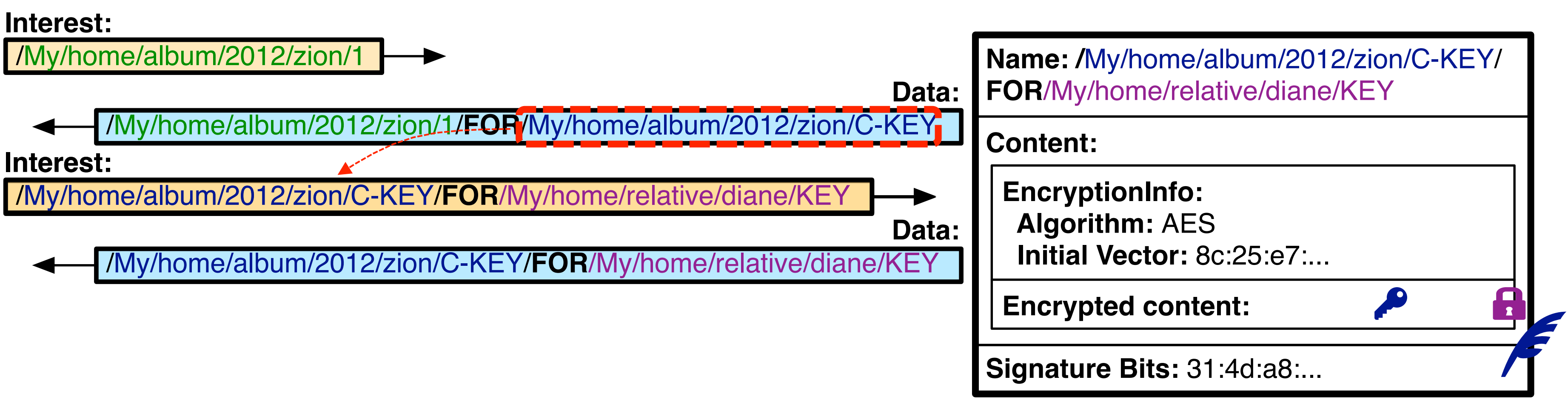
  /[content_name]/FOR/[encrypt_key_name]

  - different keys lead to different copies of encrypted data
  - follow encryption key name, retrieve decryption key



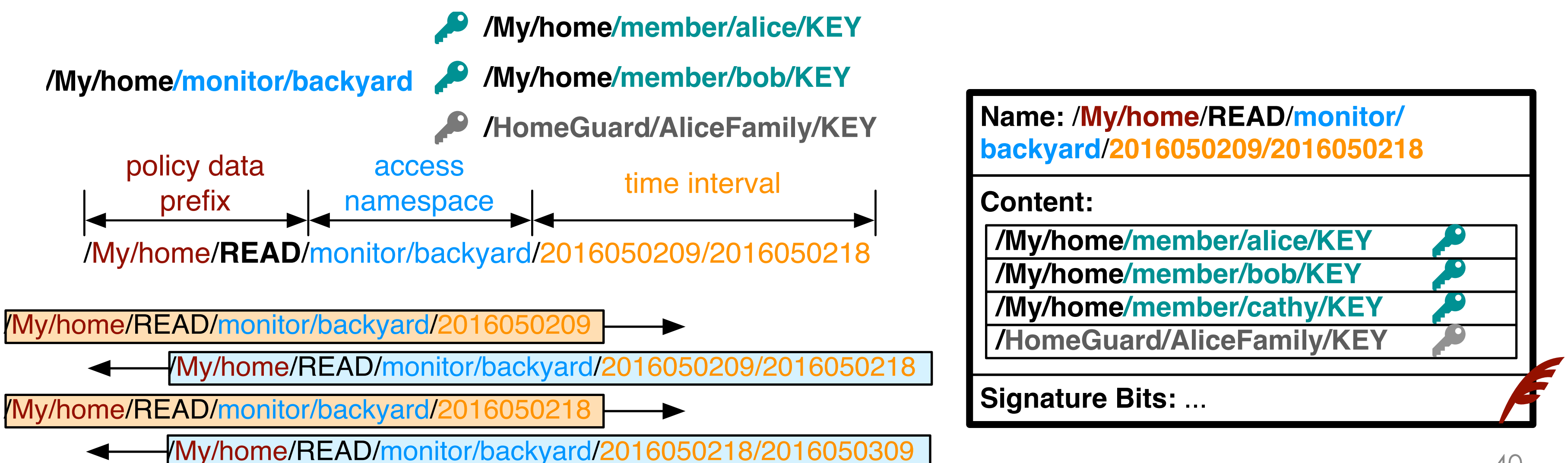
Content key distribution

- Distribute content key as encrypted data
  - encrypted using authorized consumer's public key
  - producer can publish the encrypted content key later
  - consumer can construct a decryption chain following the names



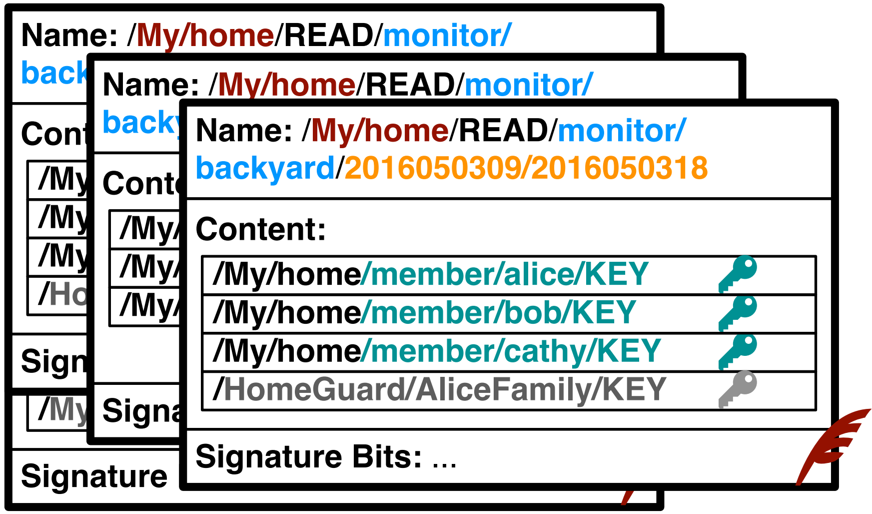
Access control policy distribution

- Some producers require updated access control policy
  - surveillance camera
- Access control policy
  - a list of (namespace, authorized consumer key set)
- Namespace owner publishes access control policy
  - producer retrieves the latest policy

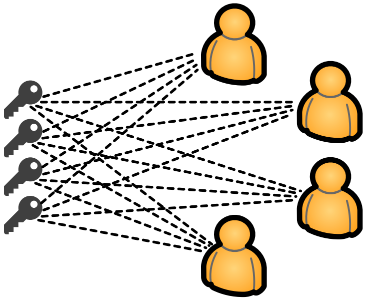


Scalability issues

- Policy retrieval overhead
  - large data packet for popular namespace
  - redundant key retrieval



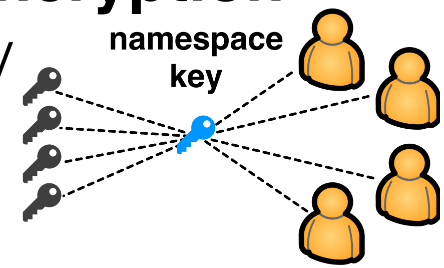
- Key encryption overhead
  - a large number of content key
  - an encrypted copy of content key for each authorized consumer
  - numbers of encrypted copies of content keys: O(mn)
    - m: number of authorized consumers
    - n: number of content keys

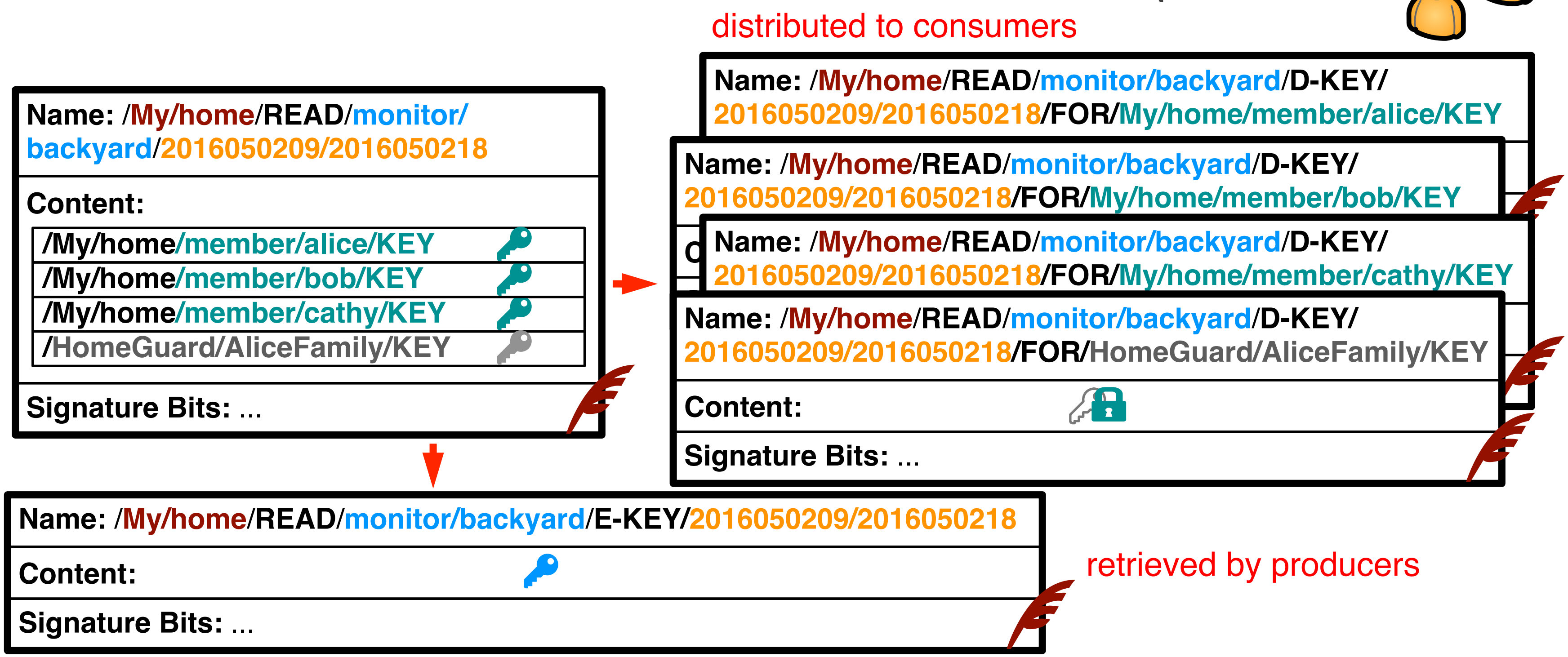


Does a producer have to know all the authorized consumers?

Namespace encryption key

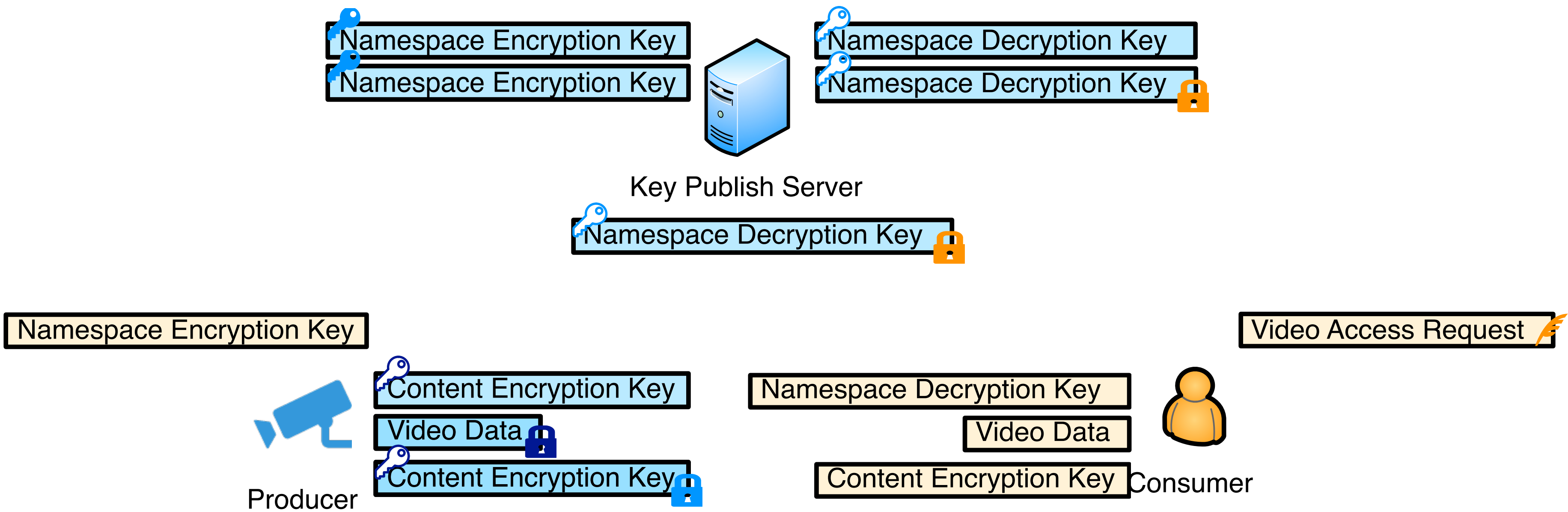
- Namespace owner publish **namespace encryption keys** instead of namespace access policy
- Number of encrypted copies: O(m+n)





Automate granting access

- Namespace owner can run a key publishing server to automate data encryption
  - validate consumer's access request using trust schema
  - generate namespace decryption key for requesting consumer



Implementation

- Available in all the NDN platform libraries
  - ndn-group-encrypt:
    - http://github.com/named-data/ndn-group-encrypt/
  - NDN-CCL
    - http://named-data.net/codebase/platform/ndn-ccl/
- Powers data access control in:
  - NDNfit: health data sharing over NDN
  - EBAMS: building management system over NDN

Summary

- Data-centric confidentiality is a **decryption key distribution** problem
  - control access by publishing encryption/decryption keys
- Key name specifies access at **fine granularity**
  - automate data encryption
- Indirected encryption enables **scalable** key distribution

Conclusion

- Data-centric security model enables flexible data communication model
  - reduced dependency on the data containers and channels
- Usability is critical to any security solution
  - developers need high-level abstraction
  - automation minimizes developer's workload
- Expressive names enables usable security in NDN
  - provide sufficient context and fine granularity for least privilege
  - naming pattern can represent flexible trust models and automate authentication & encryption

Future work

- Trust schema bootstrapping
- Robust timestamp service
  - multiple instances
  - failure recovery
- Enable name confidentiality

List of publications

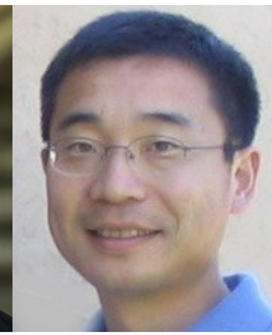
- Journal and conference papers
    - Y. Yu, A. Afanasyev, D. Clark, kc claffy, V. Jacobson, and L. Zhang, "Schematizing Trust in Named Data Networking," Proc. of ACM ICN, 2015.
    - A. Afanasyev, Z. Zhu, Y. Yu, L. Wang, and L. Zhang, "The Story of ChronoShare, or How NDN Brought Distributed Secure File Sharing Back," in Proc. of IEEE MASS, 2015.
    - Y. Yu, D. Wessels, M. Larson, and L. Zhang, "Check-R: A New Method of Measuring DNSSEC Validating Resolvers," in Proc. of IEEE TMA Workshop, 2013.
    - Y. Yu, D. Wessels, M. Larson, and L. Zhang, "Authoritative Name Server Selection of DNS Caching Resolvers," in ACM Computer Communication Reviews, 2012.
- Technical reports
    - W. Shang, Y. Yu, R. Droms, and L. Zhang, "Challenges in IoT Networking via TCP/IP Architecture," Technical Report NDN-0038, 2016.
    - V. Lehman, A. Hoque, Y. Yu, L. Wang, B. Zhang, and L. Zhang, "A Secure Link State Routing Protocol for NDN", Technical Report NDN-0037, 2016.
    - W. Shang, Y. Yu, T. Liang, B. Zhang, and L. Zhang "NDN-ACE: Access Control for Constrained Environments over Named Data Networking", NDN, Technical Report NDN-0036, 2015
    - Y. Yu, A. Afanasyev, and L. Zhang "Name-Based Access Control", Technical Report NDN-0034, 2015
    - Y. Yu "Public Key Management in Named Data Networking", Technical Report NDN-0029, 2015
    - Y. Yu, A. Afanasyev, Z. Zhu, and L. Zhang "An Endorsement-based Key Management System for Decentralized NDN Chat Application", Technical Report NDN- 0023, 2014
    - Y. Yu, J. Cai, E. Osterweil, and L. Zhang "Measuring the Placement of DNS Servers in Top-Level-Domain" Technical Report, May. 2011

Special Thanks To Collaborators


Lixia Zhang


Van Jacobson


Alex Halderman


Beichuan Zhang

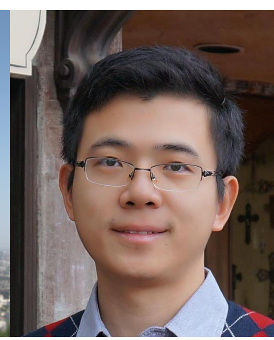

Lan Wang


David Clark

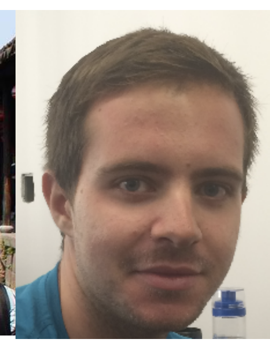

kc claffy


Alexander Afanasyev


Zhenkai Zhu


Wentao Shang


Haitao Zhang


Spyridon Mastorakis

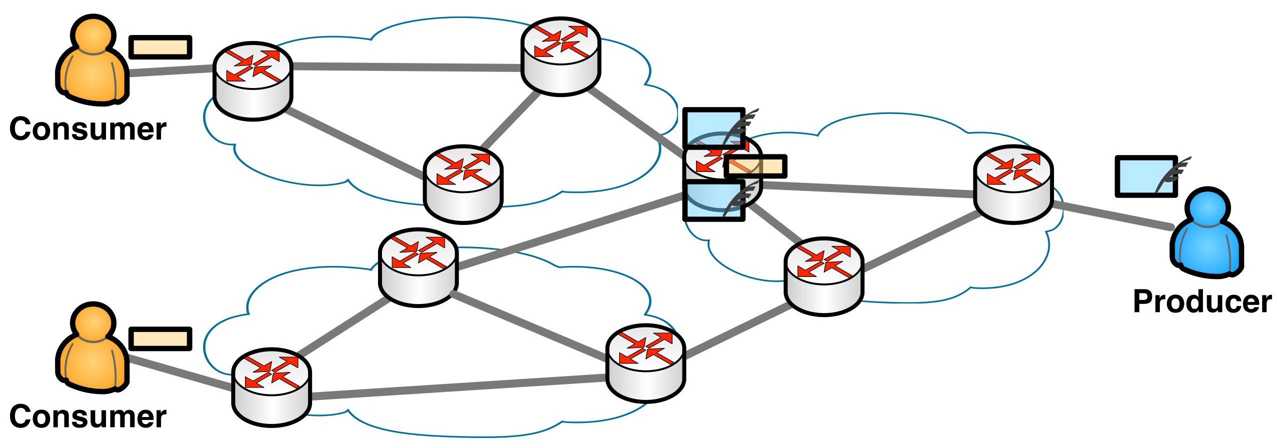

Qiuhan Ding


Prashanth Swami

End

NDN Overview

- Native multicast
  - Interest for the same data can be merged



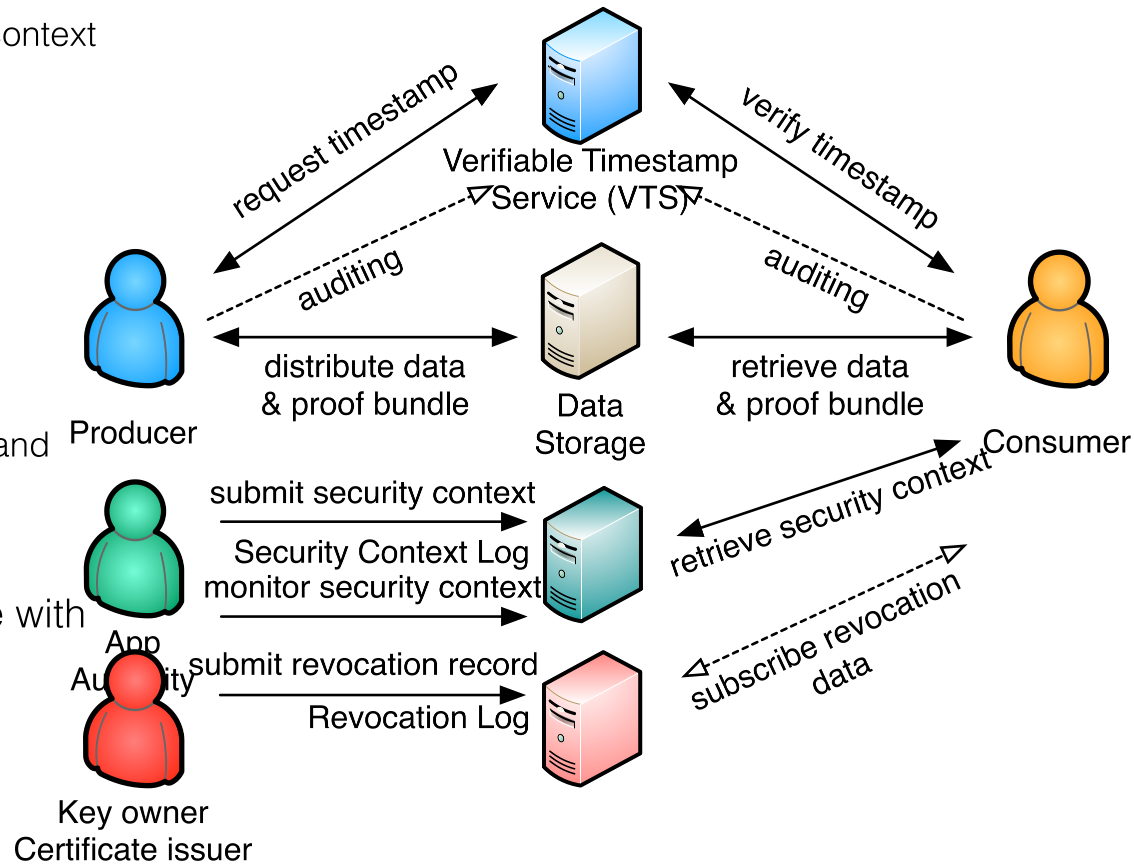
Data-centric security & NDN

- Named Data Networking (NDN)
  - data-centric communication primitives
  - retrieve data by name rather by host

- NDN enables data-centric security
  - per-packet signature
  - hierarchical naming
    - security context
    - least privilege
  - efficient key distribution

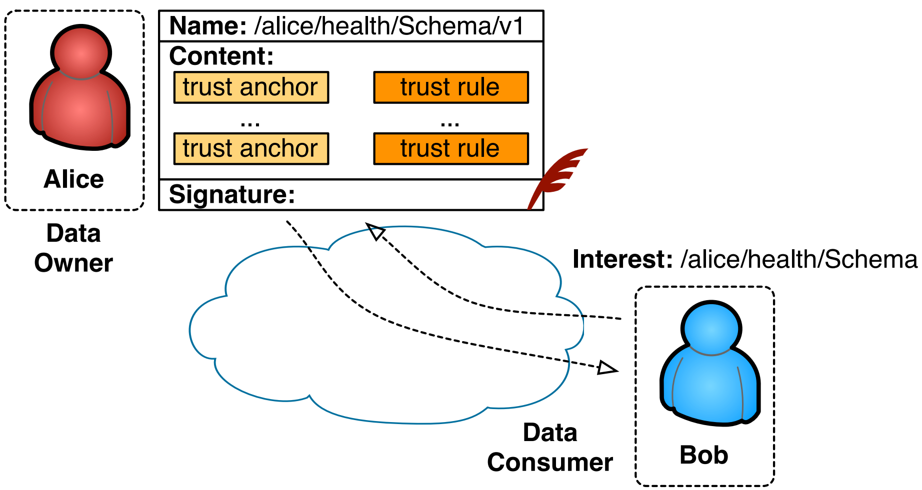
SigLogger Overview

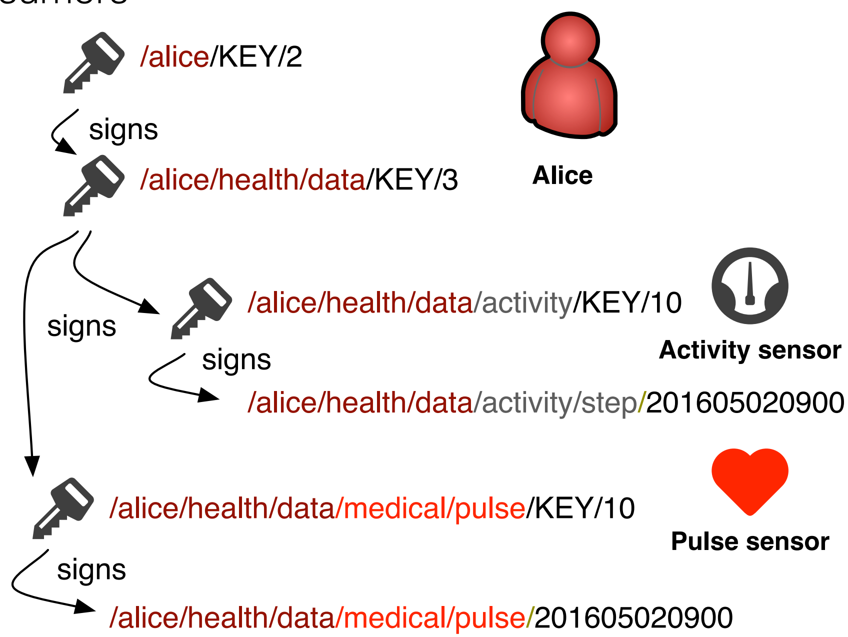
- Security Context Log
  - record security context over the time
    - trust schema
  - assure only one valid version of context at any time point
  - secure through publicity
- Revocation Log
  - record revocation over the time
  - promptly distribute revocation information to consumers
- Verifiable Timestamp Service
  - provide existence proof of data (and keys)
  - untrustworthy but auditable
- Producer distributes proof bundle with data
  - timestamp of data
  - intermediate keys
  - timestamp of keys



Signing-based write access

- Key name represents capability
  - capable of producing data under a namespace
  - capable of delegating the write access of a sub-namespace to others
  - signing key hierarchy
- Express write access control policy as trust schema
- Distribute trust schema as data
  - published by data owner retrieved by consumers





Append-only timestamp service

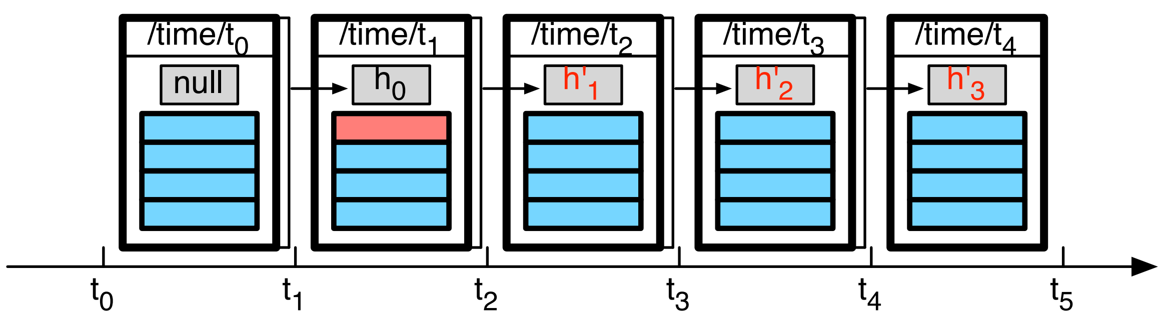
- Chaining timestamp data by hash
  - each timestamp data fixes all the previous timestamp bundle

$$h_k = H(TB_k|h_{k-1})$$

- Consistence verification

No way to know whether the timestamp service is honest about $[t_i,t_j)$

  - any modification before t_i will be detected

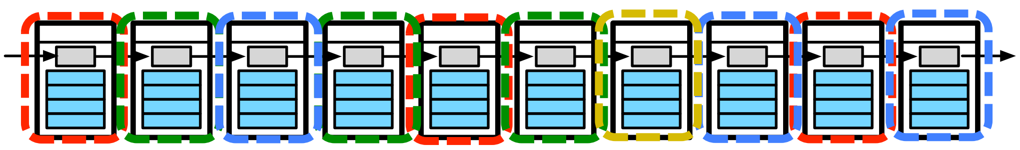


Public auditing

- Easy to catch misbehavior if
  - the consistency of each timestamp bundle is checked by at least one consumer

- Each consumer verifies consistence occasionally
- A lot of consumers collectively audit the single timestamp service

- How to minimize verification overhead



Use Case Example

- Alice obtains a timestamp for her thesis
  - also for keys if not timestampped
  - distribute data with keys and timestamps
- Bob verifies the existence of keys using timestamp
  - verifies data using keys