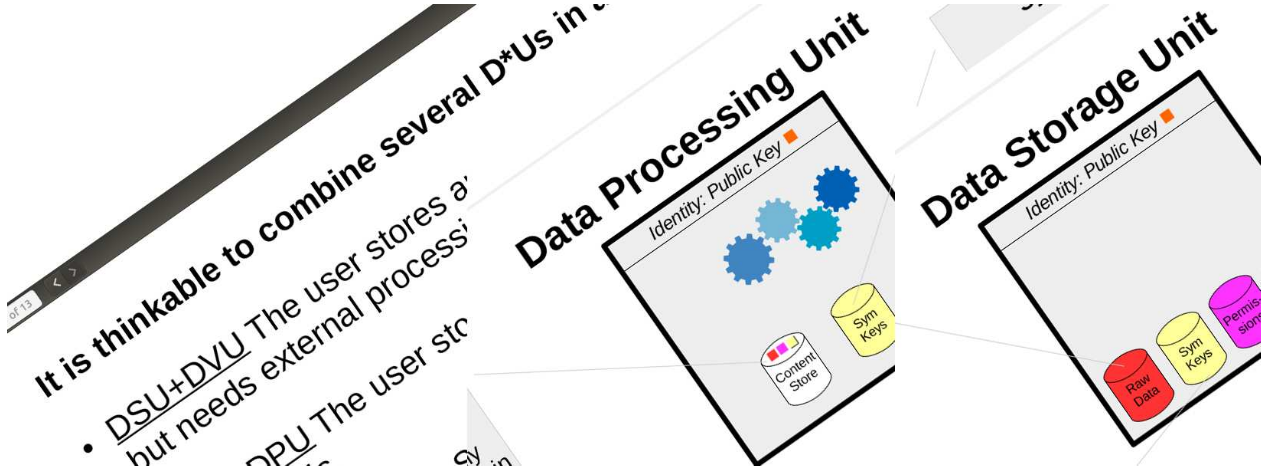
Securing NFN-style data flow processing

NDN retreat, UCSD, Feb 5+6, 2015



Christian Tschudin
with contributions from Claudio Marxer, University of Basel

Overview

1. Named Function Networking (NFN) in two slides
2. Problem statement (NDNex)
3. A blueprint for chaining Data Processing Units
4. “Deliberate Data Decoherence” : a new goal?

1) Named Function Networking (1/2)

Not a definition, but a first order approximation of NFN:

NFN = an ICN network, where Interests carry

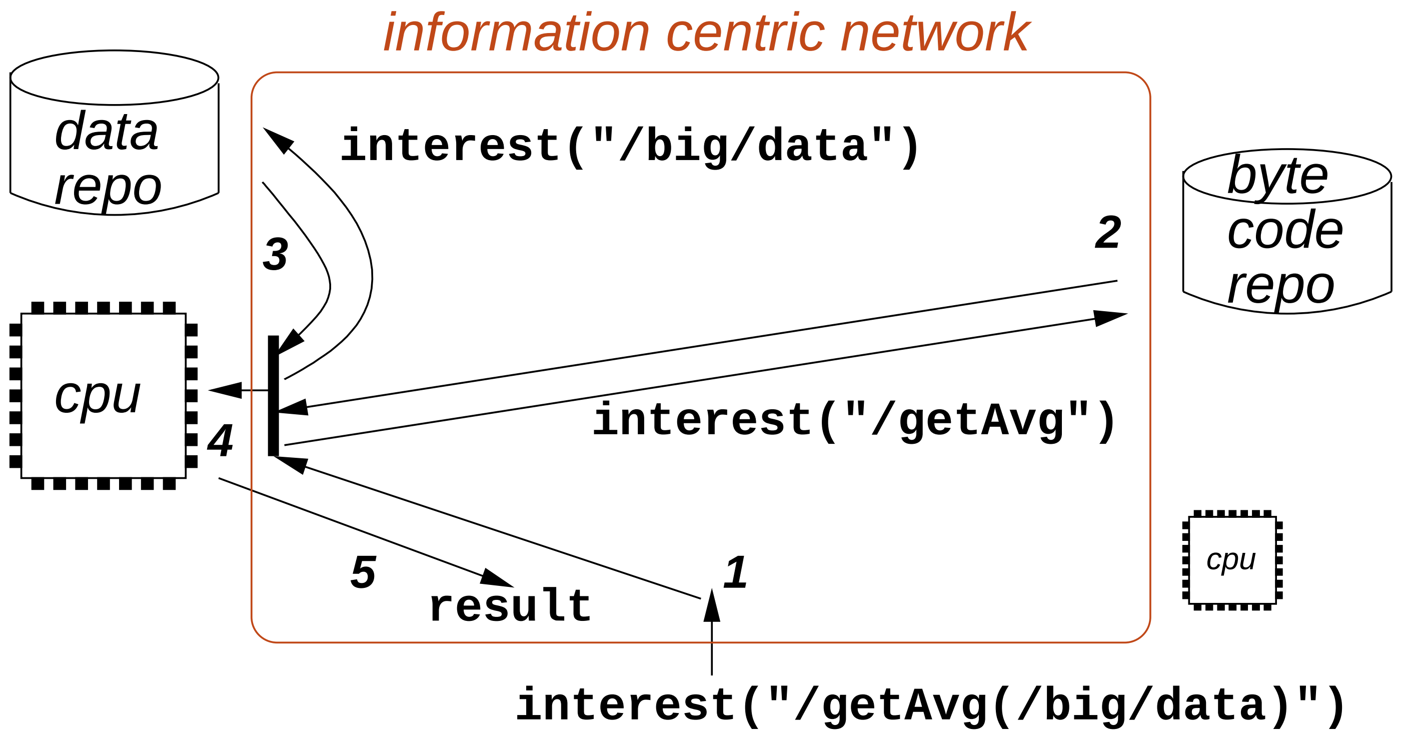
two or more names

which the network is supposed to “reduce”.

Compare with: NDN “retrieves” by single name, pure transport

1) Named Function Networking (2/2)

Three tasks: Locate data, fct and exec place; Run; Collect (and cache)



The net does **not** execute: NFN only orchestrates the computation
(think “query planning” in data bases before the actual table access)

2) Problem Statement (NDNex)

How to **selectively grant access** to private data? (doctor, insurance company, runner friends), variety of front-ends (console, alert systems), changing over time

- **Native NDN is access agnostic**: just present the name
  - need to ship encrypted data only
  - need encrypted names (often containing parameters)
- Ohmage: a) DataStorageUnit maps to NDN-Repo,
  but b) how to implement DataProcessingUnits-over-NDN?
- Approach: use **NFN for interposing access functions**
  – typically a chain of data processing functions (filters)

Assumptions: PKI, asymm keys identify the roles (DSU, DPU, DVU) which can be instantiated multiple times and location-independently.

3) A Blueprint for Chaining DPUs

See the next four slides.
This is work in progress (Claudio Marxer, UofBasel)

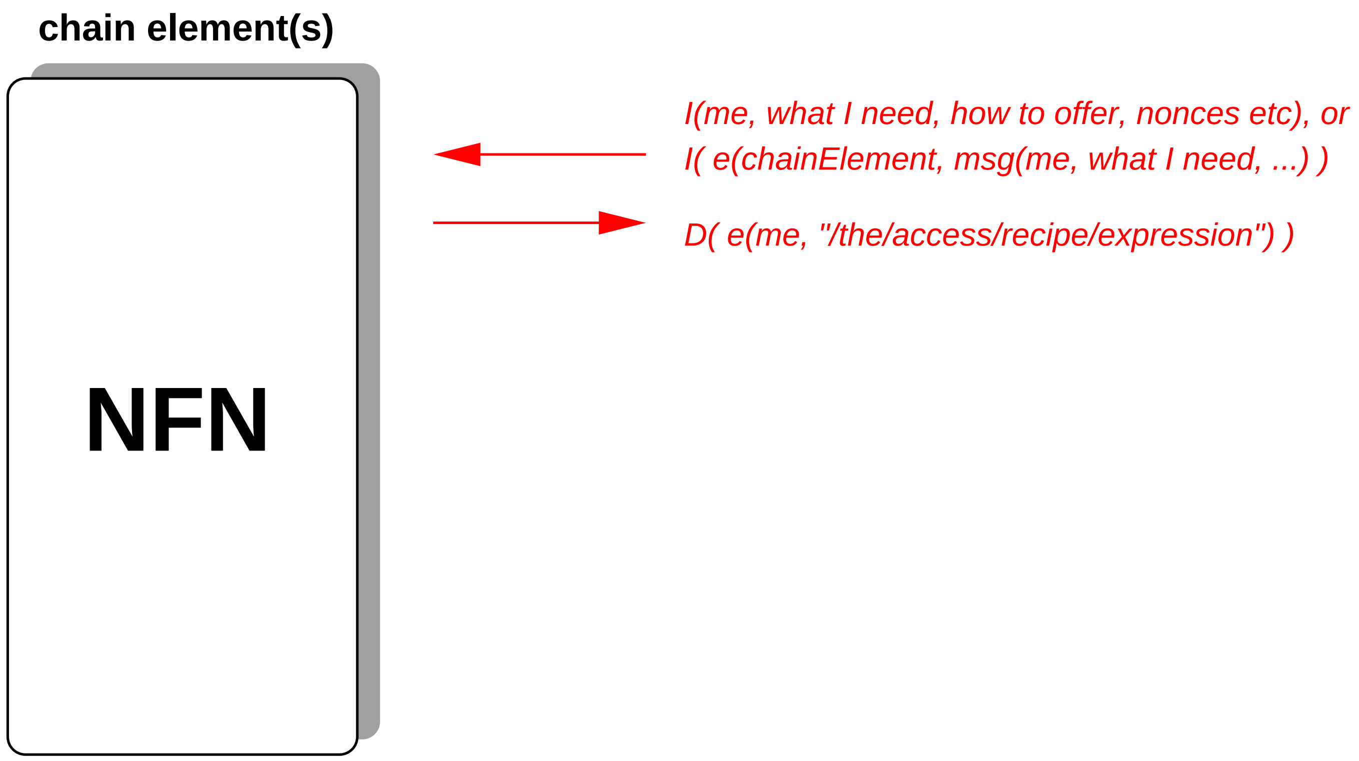
- First insight: Envisage three independent activities. Client must

  a) request access, b) request production, c) request keys

- "retrieve-the-named-data" is replaced by:
  a) client must "describe and justify" the request, then
  some access control logic (NFN) to investigate, decide
  b) decision comes in form of a stub/proxy, must be invoked (NFN)
  c) independent key retrieval (permitting NDN to do caching)

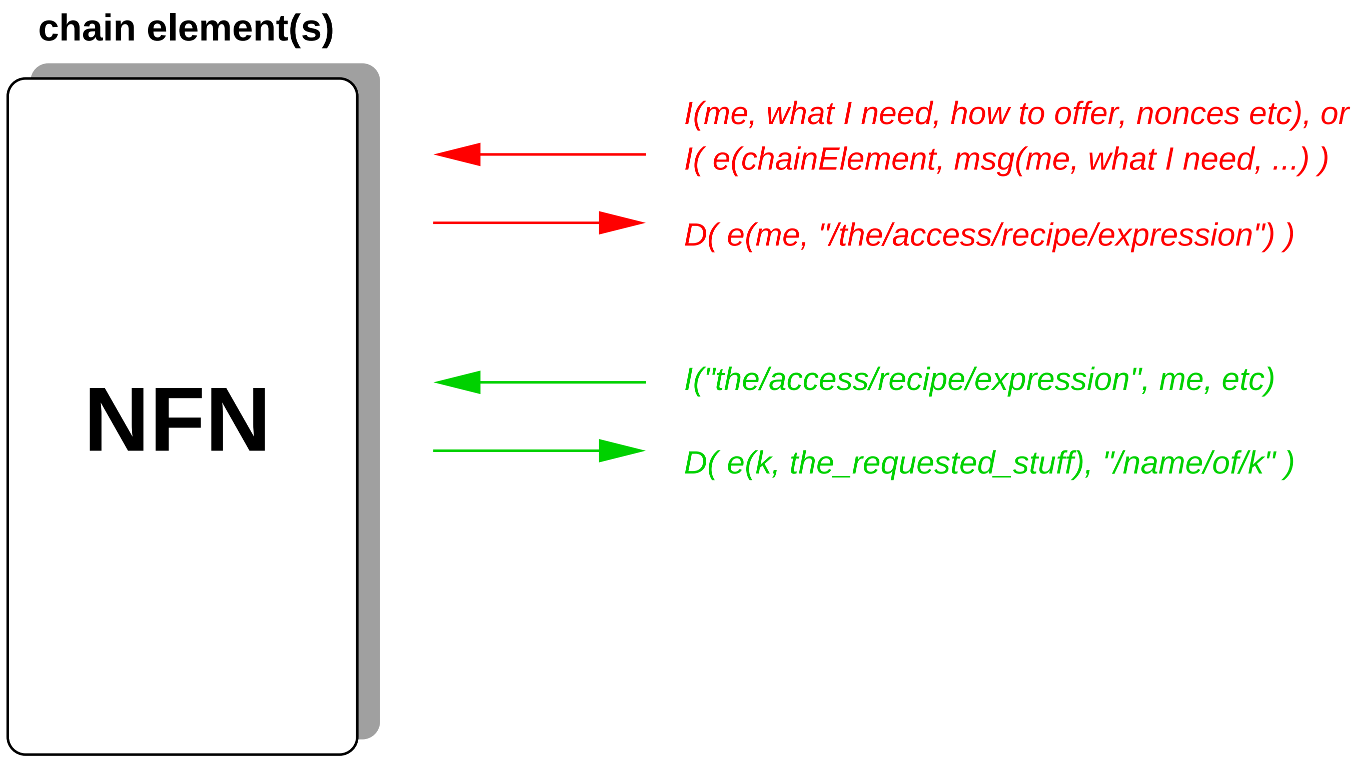
NFN-over-NDN: activities to use single-name NDN primitives.

3) Securing a Data Processing Unit Chain (1/4)


chain element(s)
NFN
I(me, what I need, how to offer, nonces etc), or
I(e(chainElement, msg(me, what I need, ...))
D(e(me, "/the/access/recipe/expression"))

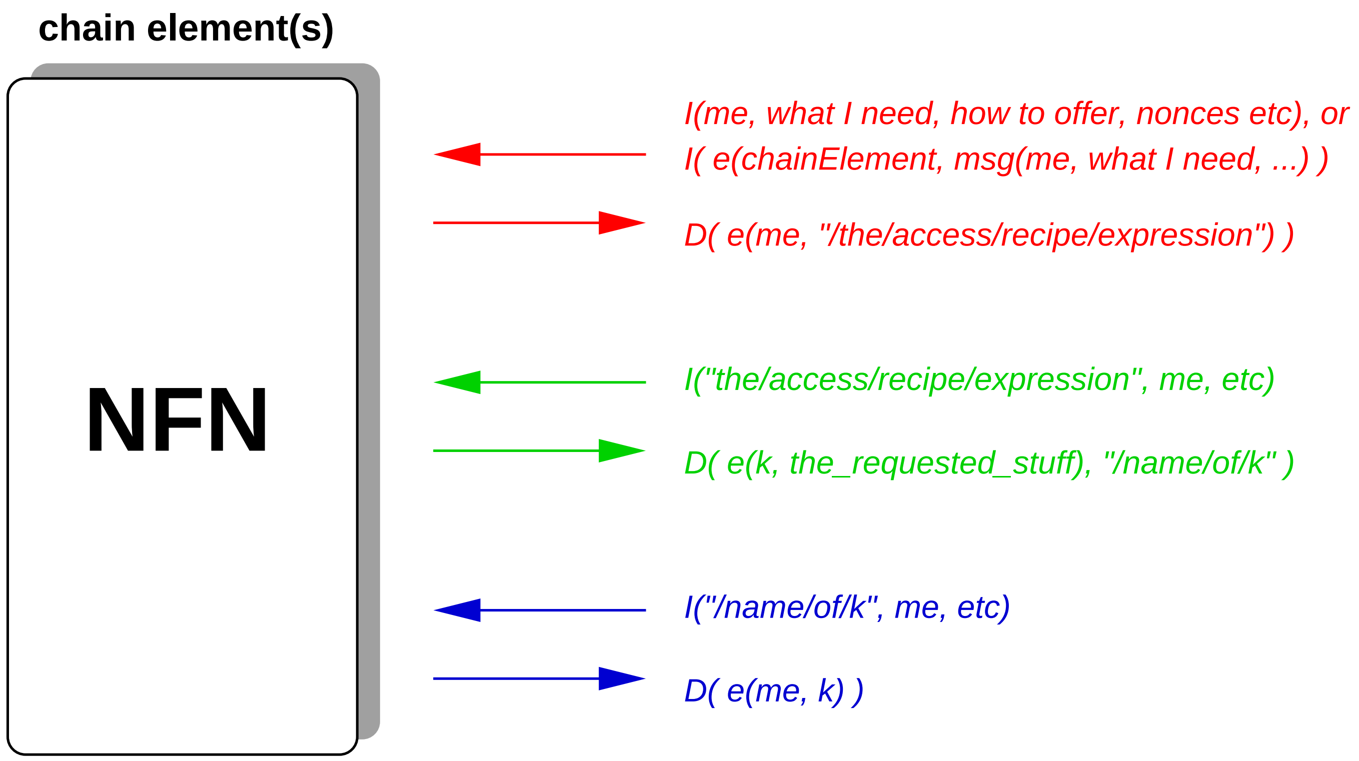
Requests are not mapped to names, but “recipes”
(with plain names being a special case)

3) Securing a Data Processing Unit Chain (2/4)


chain element(s)
NFN
I(me, what I need, how to offer, nonces etc), or
I(e(chainElement, msg(me, what I need, ...))
D(e(me, "/the/access/recipe/expression"))
I("the/access/recipe/expression", me, etc)
D(e(k, the_requested_stuff), "/name/of/k")

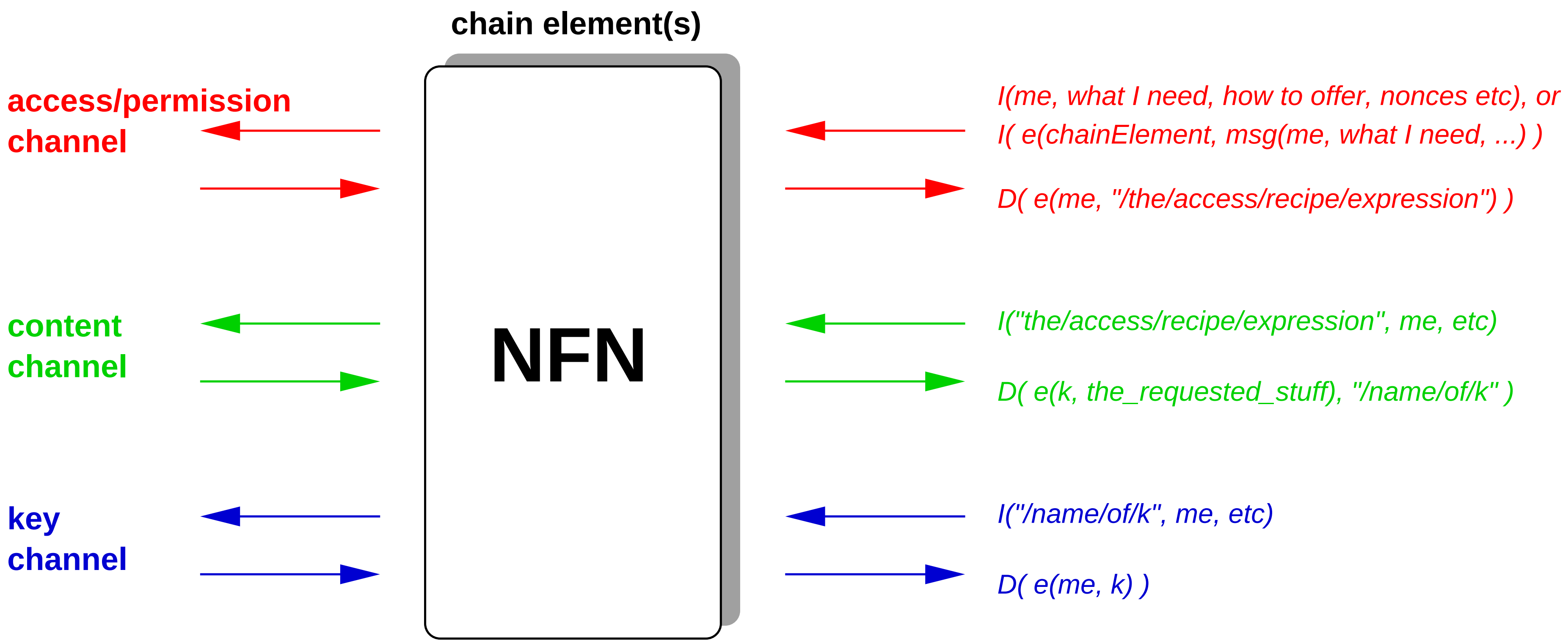
Content is accessed through a recipe, returned encrypted
(now we need that content-specific key)

3) Securing a Data Processing Unit Chain (3/4)


chain element(s)
NFN
I(me, what I need, how to offer, nonces etc), or
I(e(chainElement, msg(me, what I need, ...))
D(e(me, "/the/access/recipe/expression"))
I("the/access/recipe/expression", me, etc)
D(e(k, the_requested_stuff), "/name/of/k")
I("/name/of/k", me, etc)
D(e(me, k))

Content keys are accessed through names, returned encrypted

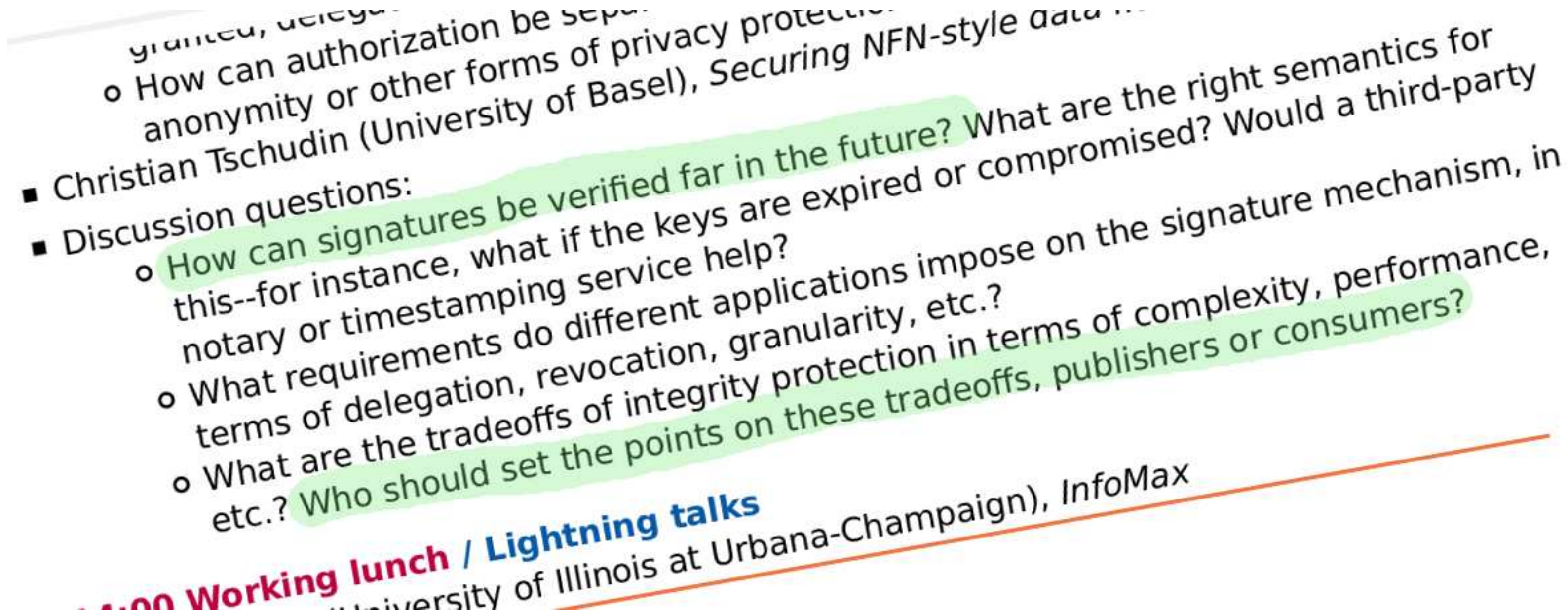
3) Securing a Data Processing Unit Chain (4/4)


chain element(s)
access/permission channel
content channel
key channel
NFN
I(me, what I need, how to offer, nonces etc), or
I(e(chainElement, msg(me, what I need, ...))
D(e(me, "/the/access/recipe/expression"))
I("the/access/recipe/expression", me, etc)
D(e(k, the_requested_stuff), "/name/of/k")
I("/name/of/k", me, etc)
D(e(me, k))

Summary: access control through three logical "channels", where keys hide the bits, recipes hide the names

4) Speculating beyond the current NDN architecture

- Jeff Burke (?) listed tough questions,
  picking the one just following this talk in the agenda



granted, delega

- How can authorization be sepa
  anonymity or other forms of privacy protectio

▪ Christian Tschudin (University of Basel), *Securing NFN-style data*

▪ Discussion questions:

- How can signatures be verified far in the future? What are the right semantics for this--for instance, what if the keys are expired or compromised? Would a third-party notary or timestamping service help?
- What requirements do different applications impose on the signature mechanism, in terms of delegation, revocation, granularity, etc.?
- What are the tradeoffs of integrity protection in terms of complexity, performance, etc.? Who should set the points on these tradeoffs, publishers or consumers?

:00 **Working lunch / Lightning talks**

niversity of Illinois at Urbana-Champaign), *InfoMax*

- Signature verification needs clarification, is a pain.
  Should the network do more work, promise more?

4) Speculating beyond the current NDN architecture (2/3)

If the question is *“How can signatures be verified far in the future?”*
my answer is *“Do not! Have Deliberate Data Decoherence instead”*

- Data are to survive *only if* they are part of a “keep-alive” loop which links to economics: it must be worth to preserve access
- Outside that keep-alive curation effort, decoherence must occur: data access only provided inside a small keys+time+space corridor, massive name rewriting and re-signing is good
- Repos good for persistent data, network only cares for narrow time window. Literally: net only transports content signed less than X minutes ago, not years.

Goal is to push long-term validation, revocation etc to app space.

Another word for Deliberate Data Decoherence: The net is not a historian.

4) Speculating beyond the current NDN architecture (3/3)

Where do short-lived signing keys (and certificates) fit into the picture?

context	*characteristics*	*order of magnitude*
humanity	mankind's information	yotta, accumulating
data	more names than content, names change, dynamic data	zetta
repos	long- and short-lived names	tera
NDN network	"small" set of signing keys, certificates	billions, constant
NDN node	in-memory	$<$ thousands

– there is a bit$\times$duration price for the global set of signing keys and certs.

– publishers to re-sign ("re-key") the name-content binding often (=on-demand)

– pro-active deployment of a network-wide *working set for keys*

– key channel: constant "flow of fresh keys", could do proactive push alongside rtg

Summary

- Named Function Networking:

  two or more names in an Interest msg, in-network “expression reduction”

- Secure chaining (DSU – DPU* – DVU):
  - access channel (“session establishment”)
  - content channel (“data generator”, “flow”)
  - key channel

- Key (and certificate) channel: is crucial for access ctrl
  - access to a data generators should expire fast
  - keep in-net signature validation mechanics, shorten life time of keys

    NFN =?= *“dynamic data, names and keys”*
  - opportunity to dimension the (fixed size) “key working set”

Appendix: “Working Set” (Wikipedia)

Definition:

Peter Denning (1968) defines *“the working set of information $W(t, \tau)$ of a process at time t to be the collection of information referenced by the process during the process time interval $(t - \tau, t)$”*.

Typically the units of information in question are considered to be memory pages. This is suggested to be an approximation of the set of pages that the process will access in the future (say during the next τ time units), and more specifically is suggested to be an indication of what pages ought to be kept in main memory to allow most progress to be made in the execution of that process.

Click to exit