Merging Modern Cryptography with CCN

Christopher A. Wood
www.christopher-wood.com

RIT - UCI

August 6, 2013

Agenda

1. Content-Centric Networking Fundamentals
2. Proxy Re-Encryption
3. Modern Cryptographic Schemes
4. Looking Ahead

CCN Overview

- Content-centric networking flips around the host-based model of the Internet architecture
  - *Content names*, rather than content locations, become addressable.
  - The network is permitted to store (cache) content that is in high demand
  - End result: less traffic to/from the content's original source, better usage of network resources, less latency, etc etc.

CCN Overview (continued)

How is data actually retrieved?

- A consumer C sends out an *interest* for content they desire.
- A router R_i use the information in their forwarding information base (FIB) table and data in cached in their content store (CS) to handle incoming interests:
  1. If content with the same name matches what's stored in the CS, return that content
  2. Else, store the interest in their pending interest table (PIT) (including the downstream router R_{i-1} or consumer C that made the request), and forward the request upstream to the next router R_{i+1} based on their FIB.
  3. FIBs are configured using protocol similar to OSPF
- Once the interest is satisfied in R_i, the PIT entry is cleared, the content is cached, and the data is sent downstream to C or R_{i-1}.

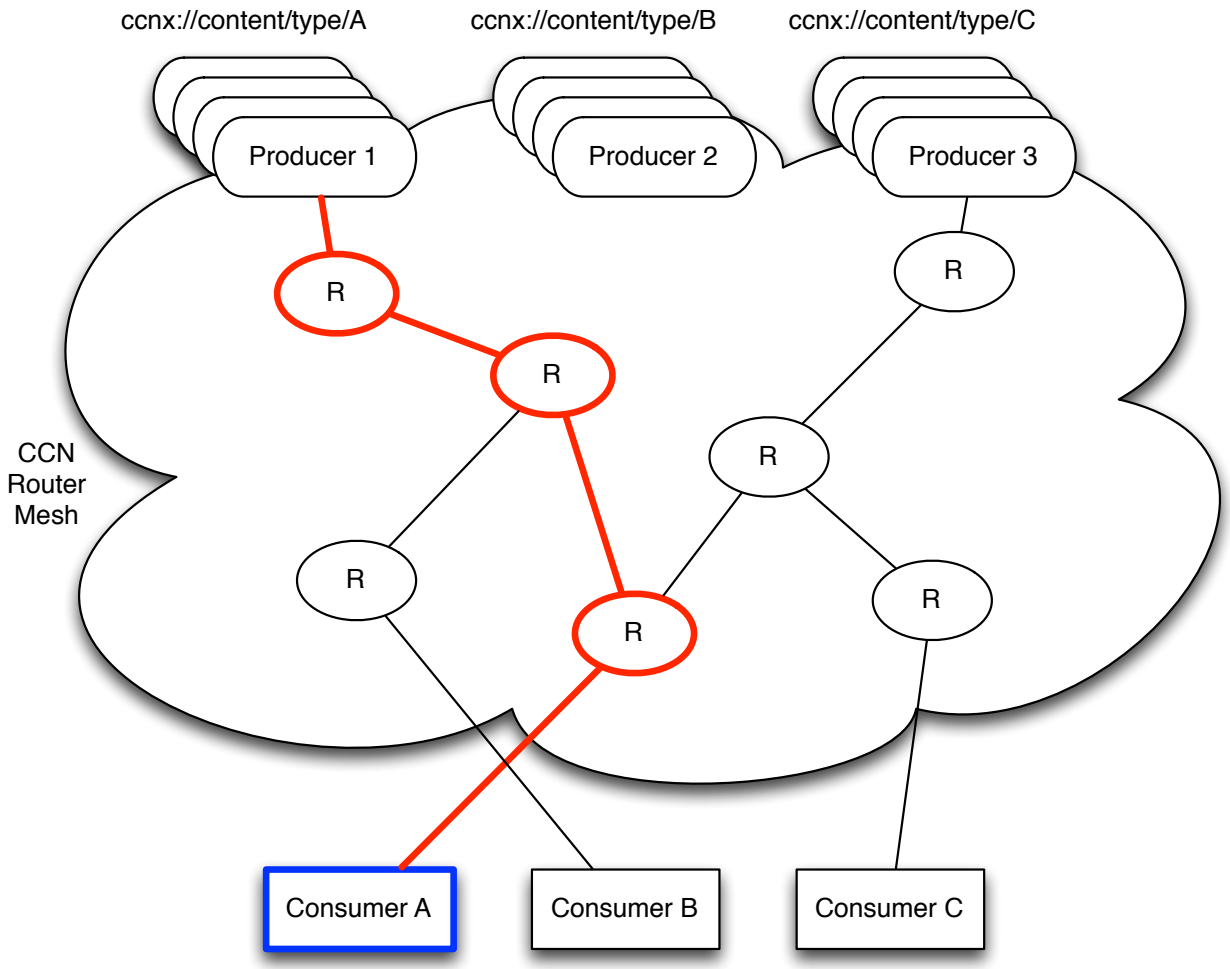
Interest Format

- Interests are similar to URLs:

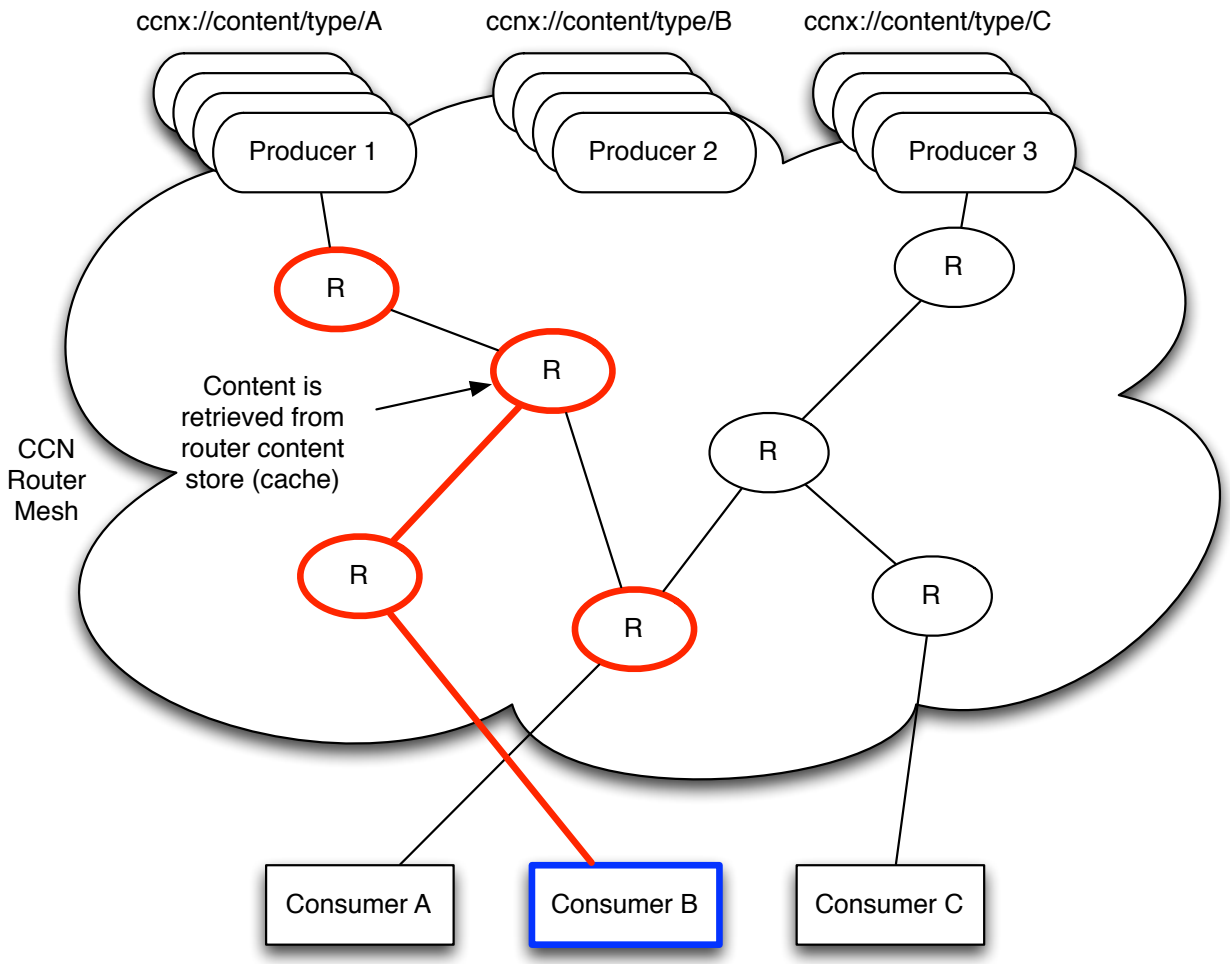
  `ccnx://rit/gccis/cs/spr/ramsey_survey`

- The / character is a delimeter that separates name *components*
- A component can be *anything*, including binary data (e.g. ciphertext)
- Interests are matched to providers in FIBs using a standard longest-prefix rule (to my knowledge, interests in CSs must match completely)

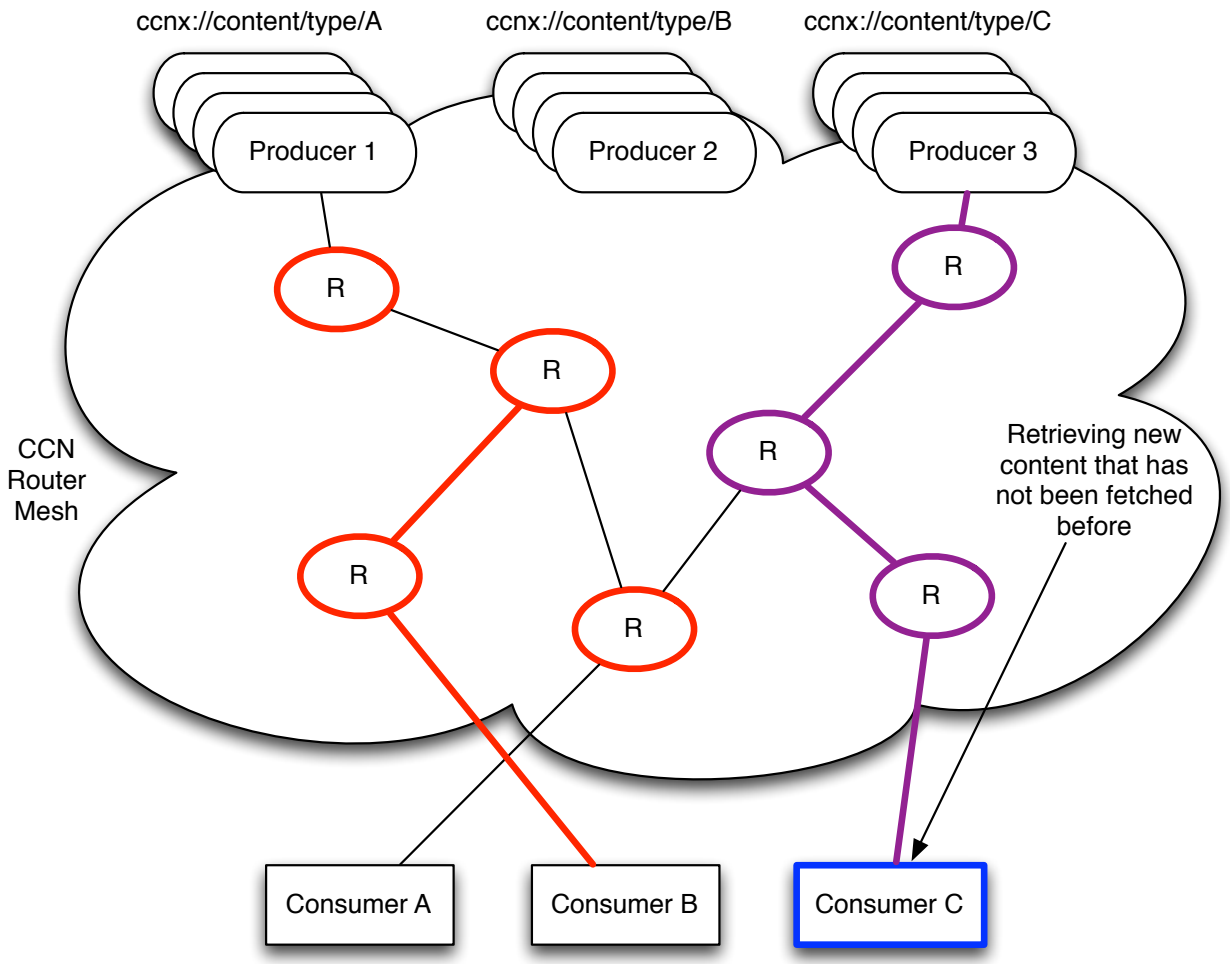
CCN in Action - #1



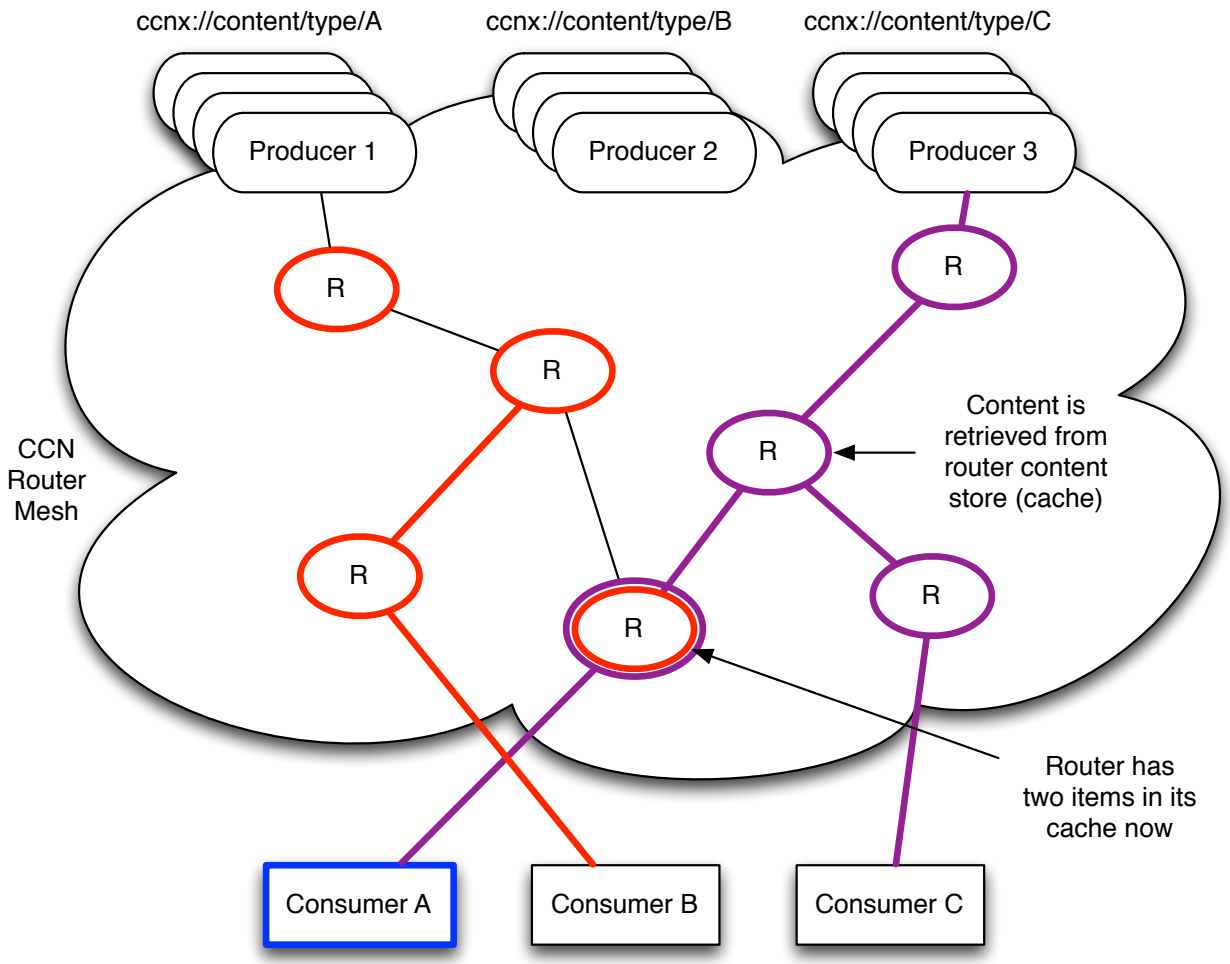
CCN in Action - #2



CCN in Action - #3



CCN in Action - #4



Notes about the Content and Cache

Content details:

- By default, *every* of data is signed by its consumer
  - Good: provides separation from security and the channel through which data is routed, and consumers can verify the validity of the content if they know the public information about the producer
  - “Bad”: In an ideal world, every router would verify the signature of every piece of content it receives from an upstream router - this is a computationally infeasible task.

Cache details:

- LRU is the default policy used in CCN (and CCNx, the open source platform)
- The network engineer is free to select cache replacement policy to suit the type of traffic

Flow Control

All traffic obeys the principle of flow control

- Interests that go “into” the network are matched with content that comes “out of” the network
- Routing decisions *are not made* based upon the content itself, but rather:
  1. Upstream traffic is routed based on the contents of the FIB (established with a routing protocol)
  2. Downstream traffic is routed based on the interface from which interests were received

Negative Aspects of CCN

Denial of Service (DoS) attacks are plentiful:

- Malicious consumers can send out fake/invalid interests (i.e. those that won't match an entry in any FIB) that steal the router's computational resources
  - There are solutions, such as exponential backoff heuristics to gradually accept less and less traffic on interfaces providing fake interests.
- Content stores can also be flooded with fraudulent data
  - Recall, signatures are *not* always verified at the routers
  - A malicious consumer can produce content that match an interest in a router's PIT but that is also invalid
    - One solution: consumers verify the signature, realize the data is invalid, and re-send an interest by specifying that content be excluded from the response (forces router to request new data from legitimate producer)

Worst of Both Worlds

Two-pronged attacks: malicious producers and consumers

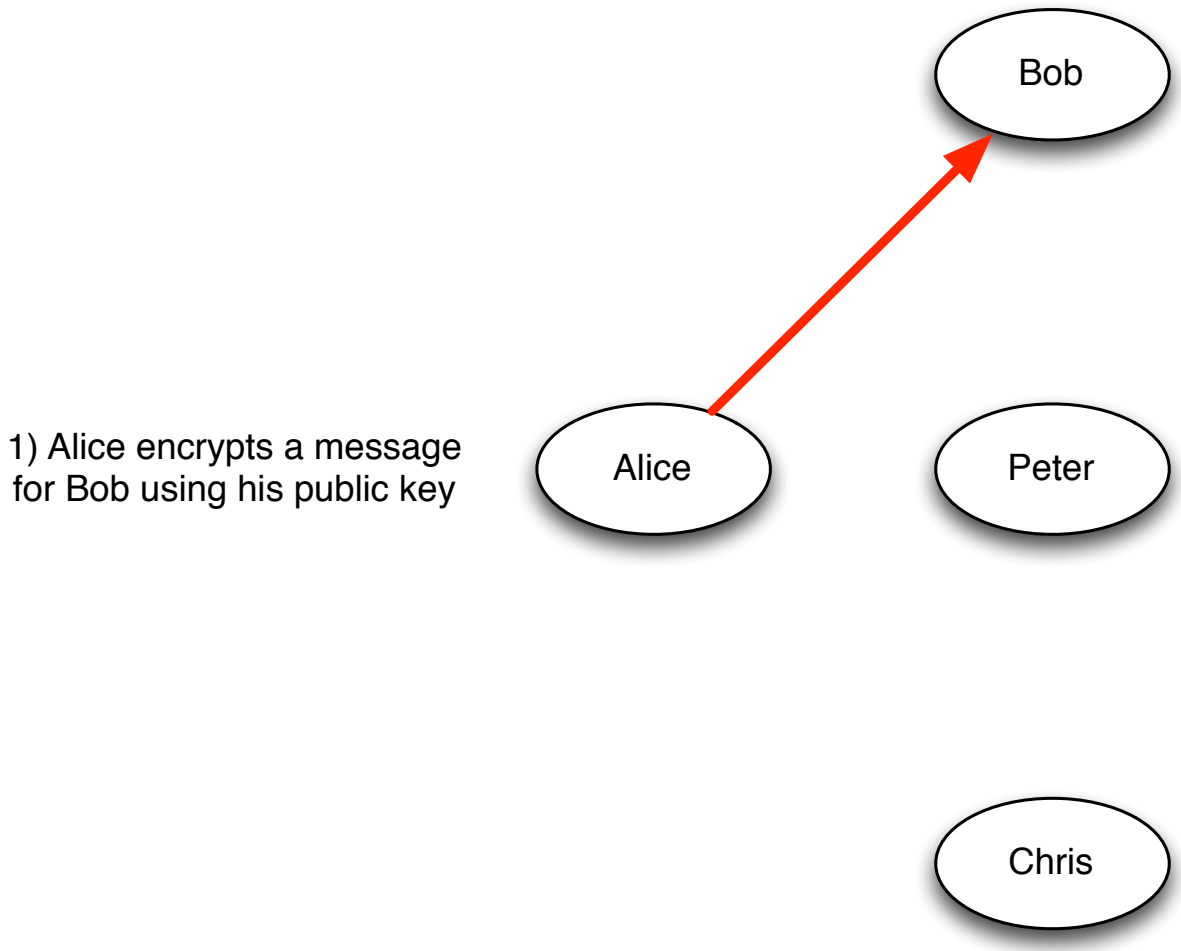
- Attackers coordinate to flood with both invalid interests and valid (but fraudulent!) returned content that will be stored in the CS
- This leaves no room for legitimate data needed by other honest consumers

Changing Gears

A closer look at some new cryptographic primitives

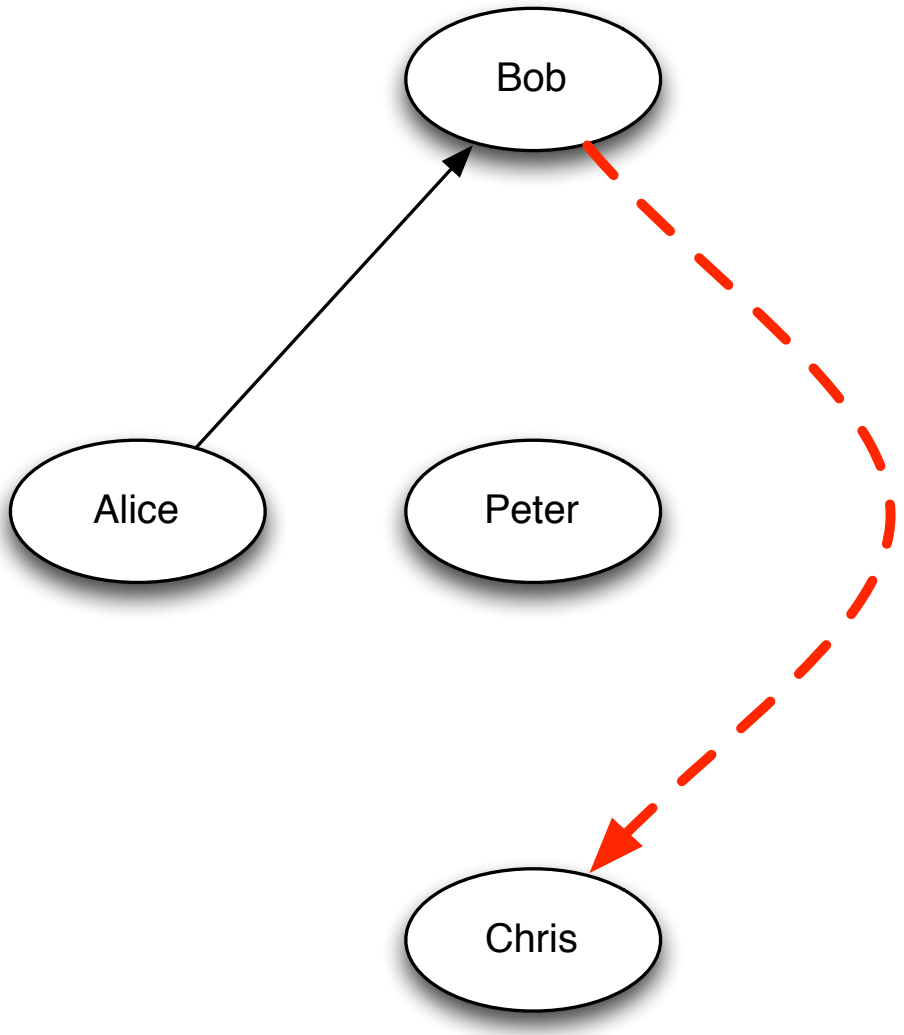
An Introduction to Proxy Re-Encryption

Consider the following scenario...



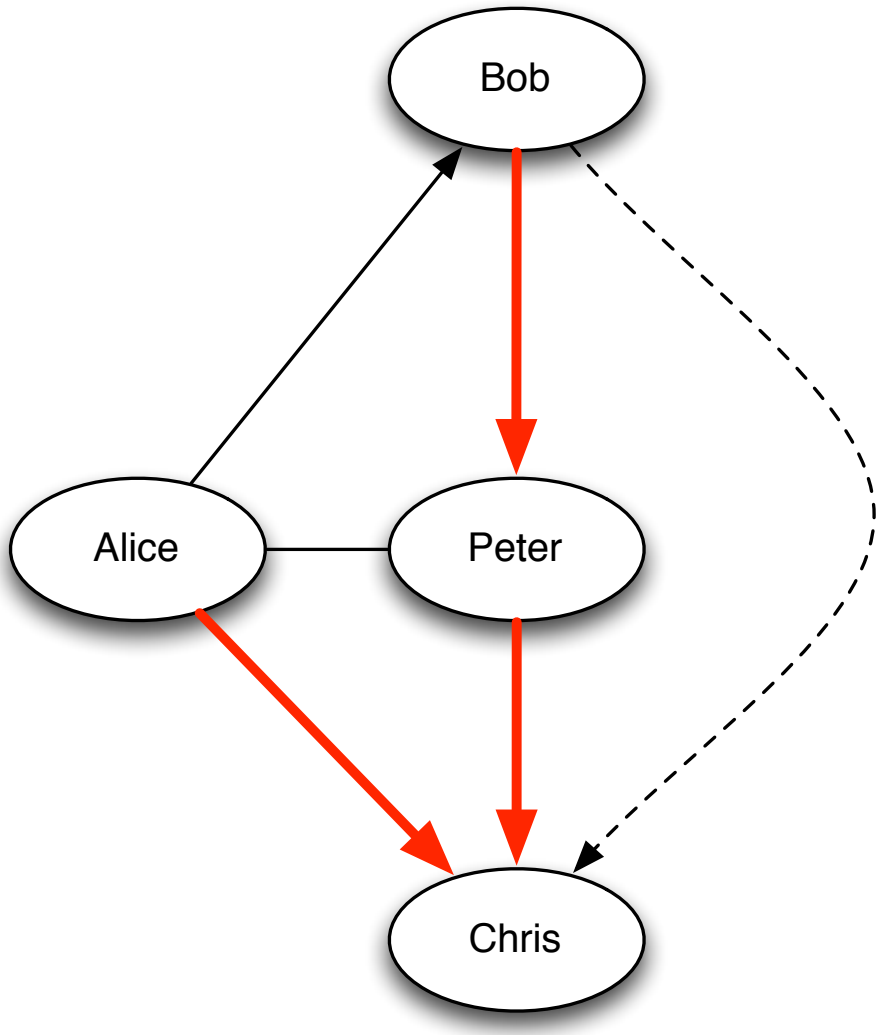
An Introduction to Proxy Re-Encryption (continued)

2) What if Bob wants Chris to read this message but has no way of getting it to him?



An Introduction to Proxy Re-Encryption (continued)

3) Chris has access to Peter and Alice, and Bob has access to Peter



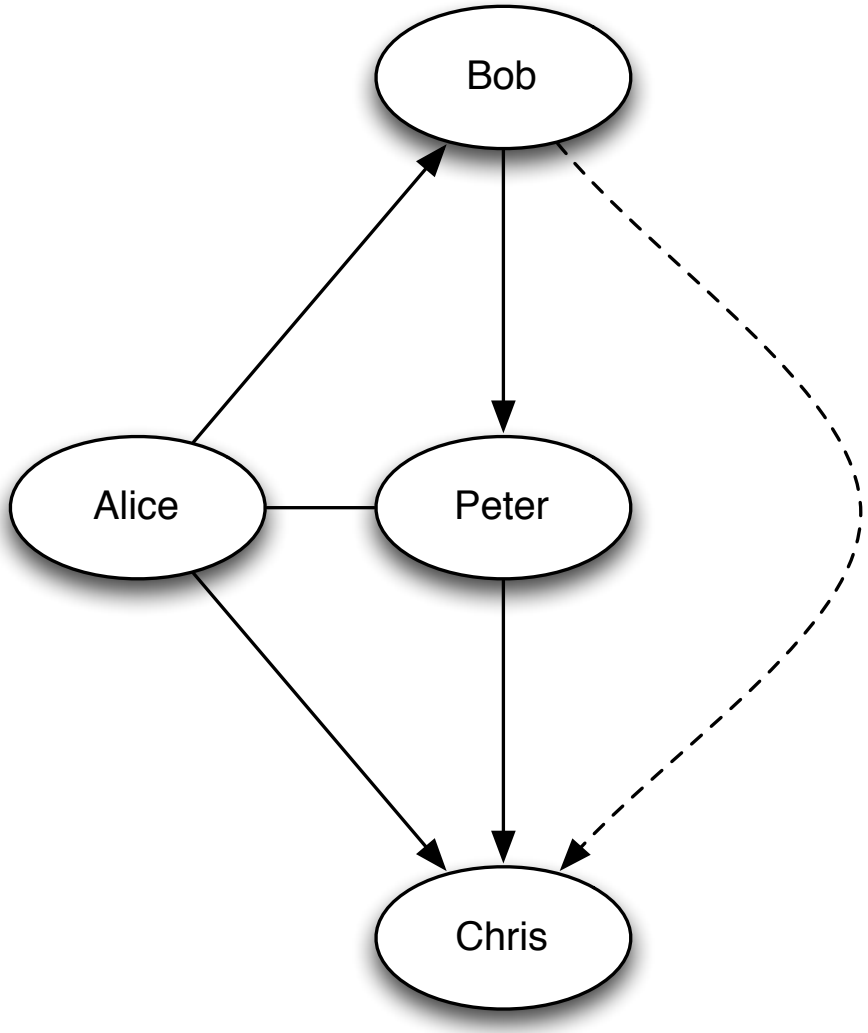
An Introduction to Proxy Re-Encryption (continued)

There are two options:

1) Alice encrypts the message again and sends it to Chris

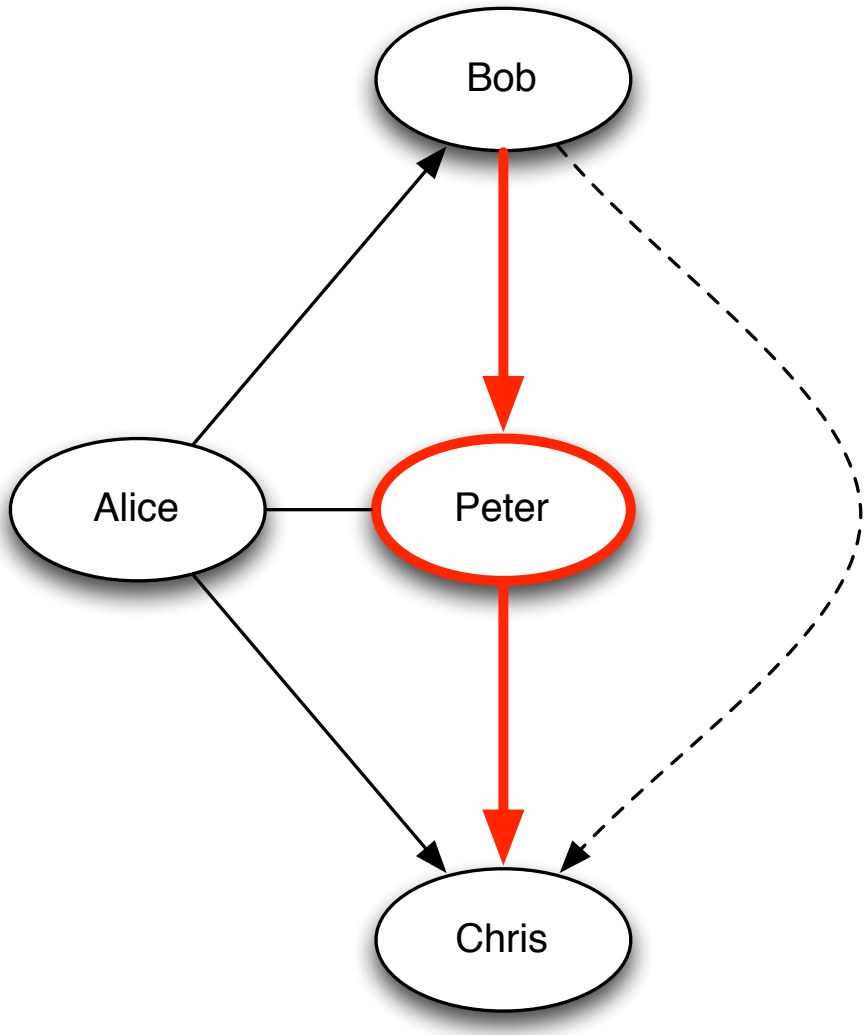
or…

2) Bob transfers the content through Peter, an untrusted proxy



An Introduction to Proxy Re-Encryption (continued)

We can use proxy re-encryption to let Peter **transform** the message to one encrypted under Alice's key without revealing any information about any secret keys or the plaintext



Proxy Re-Encryption Overview

- Proxy Re-Encryption (PRE) has been studied since the idea was first introduced at Eurocrypt in 1998.
- It can be constructed from pairings (see the Green-Ateniese identity-based PRE scheme) or clever combinations of "hashed"ElGamal public-key encryption and Schnorr signature schemes
  - Basically, the signature scheme is used to protect a token passed between parties, ElGamal encryption is used to transform plaintext masks encrypted with one public key to another, and hashes are used to hide secret keys

PRE Description

Formal description of Green and Anteniese's identity-based scheme

1. Setup - Create master secret key *msk* and public parameters *params*
2. KeyGen - Generate a secret key sk_i for user i using *params* and *msk*
3. Encrypt - Encrypt plaintext m using pk_i (i.e. the identity of user i) using *params*
4. *ReEncrypt - Reencrypt level-n ciphertext using $rk_{i \to j}$ and *params* to produce a level-$(n+1)$ ciphertext encrypted under the ID of user j
5. ReKeyGen - Generate a conversion key $rk_{i \to j}$ using sk_i, pk_i and pj_k
6. Decrypt - Decrypt a level-n ciphertext encrypted for user i using sk_i and *params*

Other Modern Schemes

EC pairings have led to *many* new and interesting pairing-based cryptographic (PBC) primitives...

- Attribute-based Encryption (KP-ABE and CP-ABE)
  - Decryption predicate: Only allow decryption of content if user possesses the attributes that meet the threshold specified by the access tree embedded in the key or ciphertext
- Hidden Vector Encryption
  - Decryption predicate: For a private key $\bar{sk} = (v_1, \ldots, v_n)$, $v_i \in \{0, 1, *\}$, and ciphertext with corresponding vector $\bar{w} = (w_1, \ldots, w_n)$, $w_i \in \{0, 1\}$, decrypt if for all $v_i \neq *, v_i = w_i$.

Modern Cryptographic Schemes (continued)

But wait... there's more!

- Inner Product Encryption
  - Decryption predicate: For a private key $\bar{sk} = (v_1, \ldots, v_n)$, $v_i \in \{0, 1, *\}$, and ciphertext with corresponding vector $\bar{w} = (w_1, \ldots, w_n)$, $w_i \in \{0, 1\}$, decrypt if

$$\sum_{i=1}^{n} v_i \cdot w_i = 0$$

- Broadcast Encryption
  - Decryption predicate: Possess a secret key corresponding to a member of the set S for which the message was encrypted (i.e. you can't decrypt if you're not a member of the intended audience) - groups are finite!

Leveraging PBC in CCN

What can we get from these PBC schemes?

- Use PBC schemes to enforce fine-grained access control on ciphertexts
- Abandon traditional PKI infrastructure and rely solely on unique identities (perhaps installed in devices before the start of a mission)
- Allow network resources such as precious bandwidth to be conserved where it matters most

Looking Ahead

Extremely fruitful possibility for future work:

Design, implement, and test more flexible content-store matching rules that are based on *content*, rather than names

What rules are useful?

- Range matching
- Substring containment
- ...

Caveat Emptor...

How can this be done at line speed?

- Unclear.
- Possibly look into:
  - Partially homomorphic encryption schemes
  - Order-preserving encryption
  - Encrypted keyword-search schemes
  - ...

I read the abstracts of these papers, nothing more. But I'm very interested to see if we can apply them in novel ways!