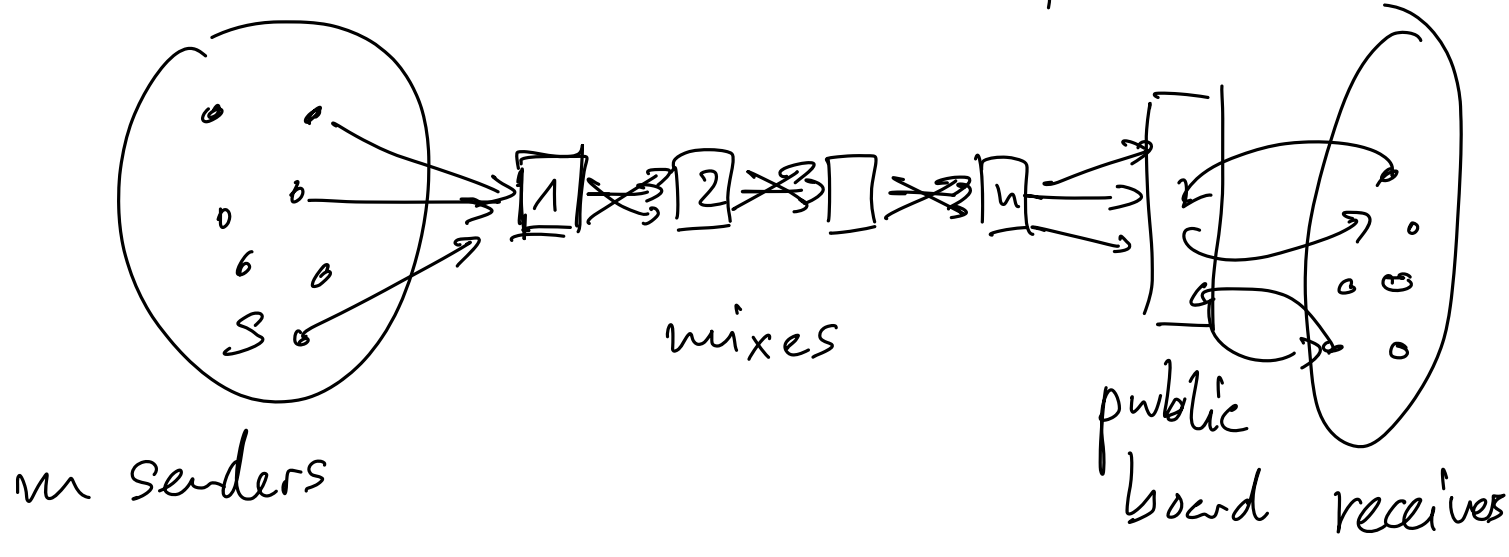
PDS, 22.12.21

9.4 Mix networks

- Ideal, but theoretical design
- Used inside larger protocols

Model

- Many senders
- Many receivers
  - Have public keys
- n communication nodes (mixes) in a sequence



- Only for datagrams
- Max. size limit

Protocol

* Sender S encrypts m to R as

$$c \leftarrow Enc(K_R, m)$$

  - perhaps re-randomizable scheme
  - pad to max length
  - send c encrypted to mix_1

* Mix protocol for mix_i:
  - Receive msg. list from mix_{i-1}
  - Decrypt msgs, permute list randomly
    re-encrypt all msg. for mix_{i+1}
  - Send list to mix_{i+1}

* Mix_n decrypts list and posts all msgs on public board

* Receivers, periodically, read all encrypted msgs from public board,

Try to decrypt each.

Output those where decryption is valid

Variations

- Add random delays
- Add dummy traffic

Properties

- Sender anonymity
  - against receiver : excellent
  - against one $(n-1)$ nodes: excellent
  - against all nodes: none
  - against global obs: excellent

- Receiver anonymity
  - against sender : excellent
  - against one $(n-1)$ nodes: excellent

- against all nodes: none
- against global obs: excellent

- Unlinkability
  - against one $(n-1)$ nodes: excellent
  - against global observer: excellent

Problems

- Limited capacity
- Only for specialized tasks,
  e.g. inside voting systems
- Not robust, active misbehavior can easily block traffic

⚠ This would leak the receive-encrypted ciphertexts to mix$_1$.

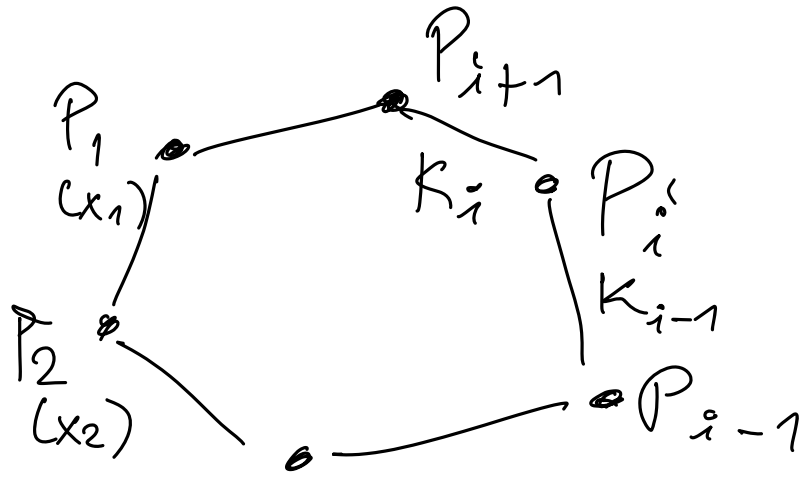
Fix this by using onion-encryption

§.5 Dining cryptographers and untraceable communication

- Cryptographers at dinner table
- Anonymous donor pays bill -
  Was it one of them or the NSA?
- Protocol for $\mathcal{P} = \{P_1, \ldots, P_n\}$
  - Each one inputs one bit x_i
    (P_i paid the bill?)
  - Goal is to compute

$$s = \bigoplus_{i=1}^{n} x_i$$

If $s=1$ then a cryptographer paid,
else NSA paid.



P_i flips one coin K_i and shows it to P_{i+1}
(indices are modulo n)

Algorithm

for $i = 1, \ldots, n$ do
  P_i chooses random bit K_i
  // only P_i and P_{i+1} see K_i

for $i = 1, \ldots,$ do
  P_i announces publicly

$$y_i \leftarrow K_i \oplus K_{i+1} \oplus x_i$$

Everyone computes

$$s \leftarrow \bigoplus_{i=1}^{n} y_i$$

Completeness

Assuming at most one x_j is 1:

$$s = \bigoplus_i y_i = \bigoplus_i k_i \oplus k_{i+1} \oplus x_i$$

$$= \bigoplus_i k_i \oplus \bigoplus_i k_i \oplus \bigoplus_i x_i$$

$$= 0 \oplus x_j$$

$$= x_j$$

Security

- Any connect set C of colluding nodes
- Graph (P, E) is the key structure
- Remove C from graph

$$(A, F) = (P \setminus C, E \setminus \{(u,v) \in E, u \in C \vee v \in C\})$$

- reduced graph of honest nodes

- Cheaters can only learn that one node in A paid

* Works for any graph

(If $|C| \geq n-1$, then sender remains hidden in the anonymity set A.)

Properties

- Sender anonymity
  - against receiver : excellent
  - against one ($n-2$) nodes: excellent
  - against all nodes: none
  - against global obs: excellent

- Receiver anonymity
  - against sender : excellent
  - against one $(n-2)$ nodes: excellent
  - against all nodes: none
  - against global obs: excellent

- Unlinkability
  - against one $(n-2)$ nodes: excellent
  - against global observer: excellent

Discussion

- Capacity limited
- Only one broadcast opportunity at the time
- If two send, the collision...
  ... have retry later.

9.6) Freenet: Anonymous censorship-resistant publishing

So far: communication (send-receive)

Now: data storage (read-write,
CRUD - create,
read, update, ~~delete~~)

Freenet design

- Peer-to-peer system: every node stores some data, for itself, for others
- Nodes communicate with "friends", people (?) known do node
- All data is stored encrypted and identified by keys
- Nodes do not know which files they store

* Content-addressable storage

- Content-Hash Key (CHK) of a file
  - is the hash of the content
  - serves a unique identifier for file

File content is additionally encrypted using K_{file}

Reference to a file is

(CHK_{file}, K_{file})

* Naming and name spaces

- Keyword-signed key (KSK)

  for a file, derived det. from a 'name' or 'keywords' deterministically

  e.g. public/private key pair using seed for RNG set to 'name'

- Anyone can insert a file under 'name'
- Anyone can retrieve files from 'name'

- Signed-subspace Key (SSK)
  - implements a name space under user's key

  User A has public key pk_A and creates SSK for a file identified by 'name' as

  $$seed \leftarrow Hash(pk_A \,||\, name)$$

  Create a publ./private key pair from seed

- Only user A can insert files under its name space
- User A publishes pk_A and name :

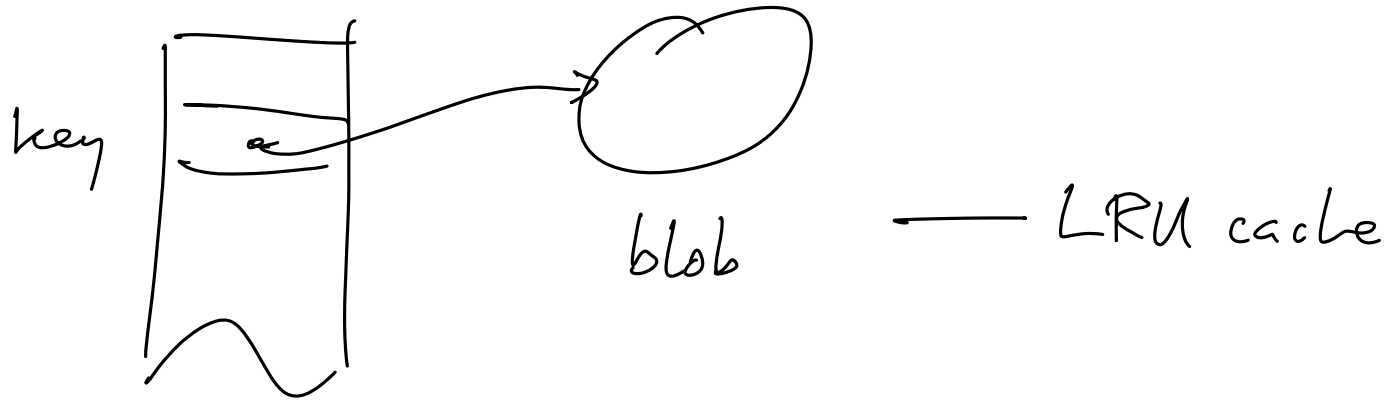
anyone with this knowledge can retrieve file

* Files are stored in 32-kB chunks and addressed indirectly

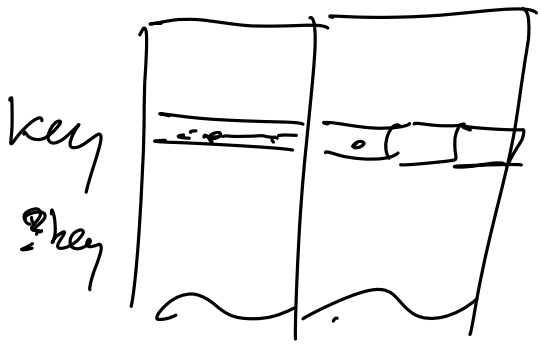
* Files are also signed under the file's key (and subspace key) to prevent tampering

Storage and retrieval protocol

- Every node stores files addressed by keys



- Every node keeps a routing table



- nodes have no complete knowledge
- To access a file, we ask repeatedly neighbors in the routing table
- Similarity of keys
- If a node has the file, sends it back
- If a node obtains a file from another node, it stores the file as well (using LRU eviction alg.)

=> File data is propagated towards nodes that retrieve it

=> File data is lost when not accessed

=> freenetproject.org