The mixminion packet format (and more)

George Danezis, Roger Dingledine, Nick Mathewson,
(Ben Laurie)

October 13, 2005

What am I going to talk about?

- **Mixminion** is a remailer:
  - Transport email anonymously.
  - Uniform(ish) size, high latency.
  - Support anonymous reply blocks.
  - Specified, implemented, and out there...
  - *BUT lacks proofs!*
- A bit of (implemented) history:
  - Type 0: `anon.penet.fi` (80s)
  - Type I: Cypherpunk remailers. (92)
  - Type II: Mixmaster (94)
  - Type III: Mixminion (03)
  - (Minx packet format – ok, this is still theory)

Use cases

- Sender anonymity: Alice sends a message M to Bob, and neither Bob nor any other third party should know that it was sent by Alice. (We call that the *the forward path*)
- Recipient anonymity: Alice can send an anonymous reply message M to Bob, and neither Alice nor any other third party knows that the message is delivered to Bob. (We call this *the reply path*).
- Bi-directional anonymity: Alice sends a message to Bob, but neither Alice knows that she corresponds with Bob, nor Bob realises that the message comes from Alice.

Anonymity is 'orthogonal' to message secrecy. And Mixminion does not provide it!

What do we assume (or not care about)?

- We care about **the cryptographic packet format.**
- Not key exchange: There is a PKI (directory service – another story) that provides the public keys of all mix servers.
- Not the mixing strategy!
- Not the network topology!
- Not the route selection!
- Not the dummy traffic policy!
- Not the the environment awareness (What?!): a mix will only know itself and the directory server.

Security requirements

Aim of a mix packet format: the message must look 'completely different' on each link! (for those that do not hold the necessary secrets).

- Anonymity despite a global passive adversary that also controls a subset of nodes on the message path, and can perform active attacks against the honest mix server network.
- A facility for secure anonymous replies, indistinguishable from other messages.
- The position of a mix one the path and the path length should be hidden from intermediate mixes and third parties.
- Tagging attacks must be eliminated.

Operational requirement

- Only parties that benefit from security properties have to use special software.

Why is it tricky?

Indistinguishable replies present a problem:

- If only forward path can use integrity checks (e.g. HMACs) on the whole packet to make sure there has been no tag. (See Anna's construction...)
- But with indistinguishable replies we cannot assume the creator of the packet knows the body of the message.
- Need to find another mechanism for preventing modification yielding any information to the attacker.

Some pictures may help...

Cryptographic primitives

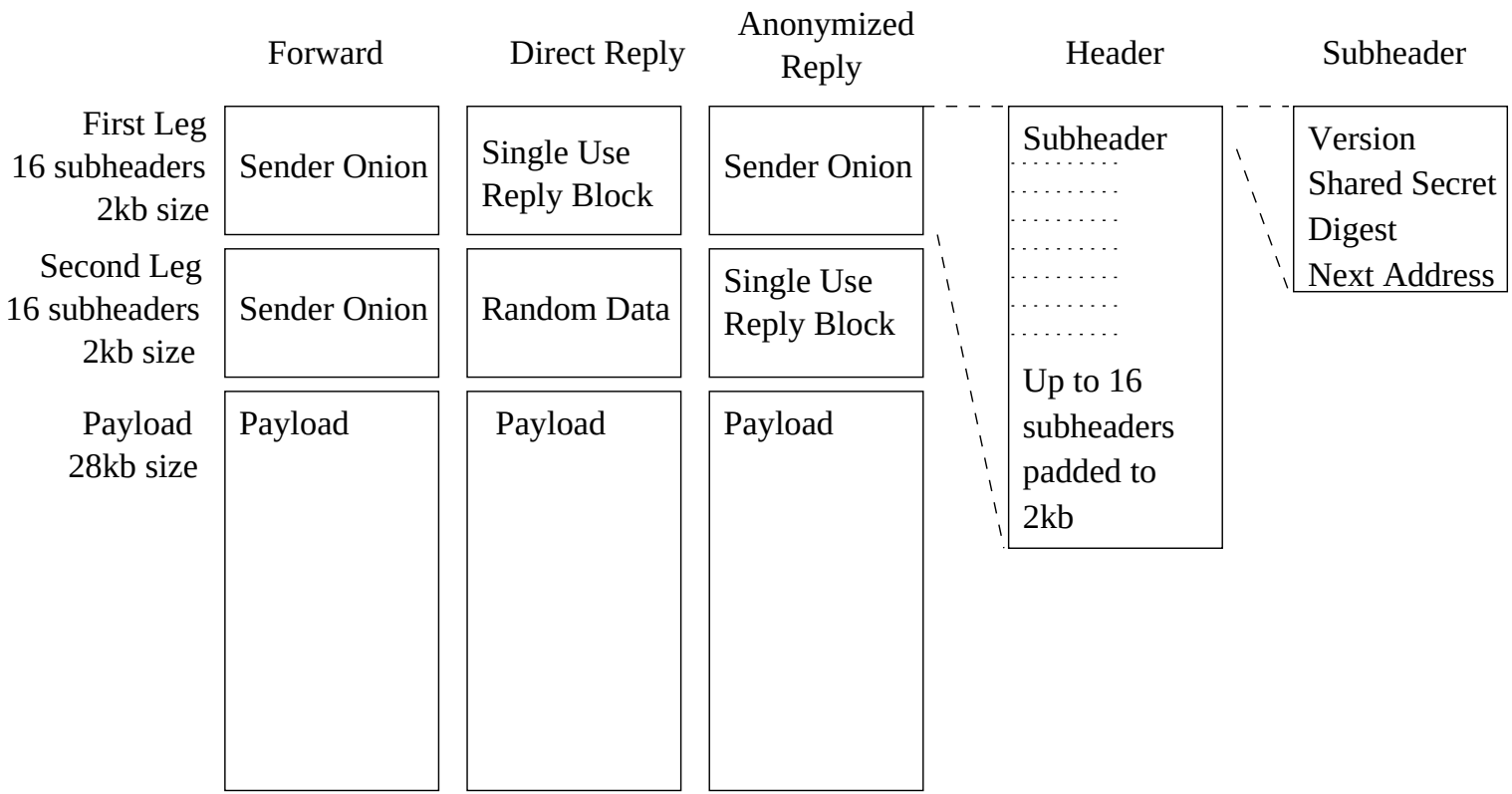
Conservative stuff:

- 1024bit RSA-OAEP (PKCS#1 Standard) – 2048bit actually
- SHA-1 for digest
- AES 128bit keys in CTR mode (with 0 IV)
- Use BEAR (or LIONESS) as a variable block size block cipher. Build from SHA-1 and AES128-CTR as a pseudo-random stream.

For more provocative design see Minx!

Packet configurations

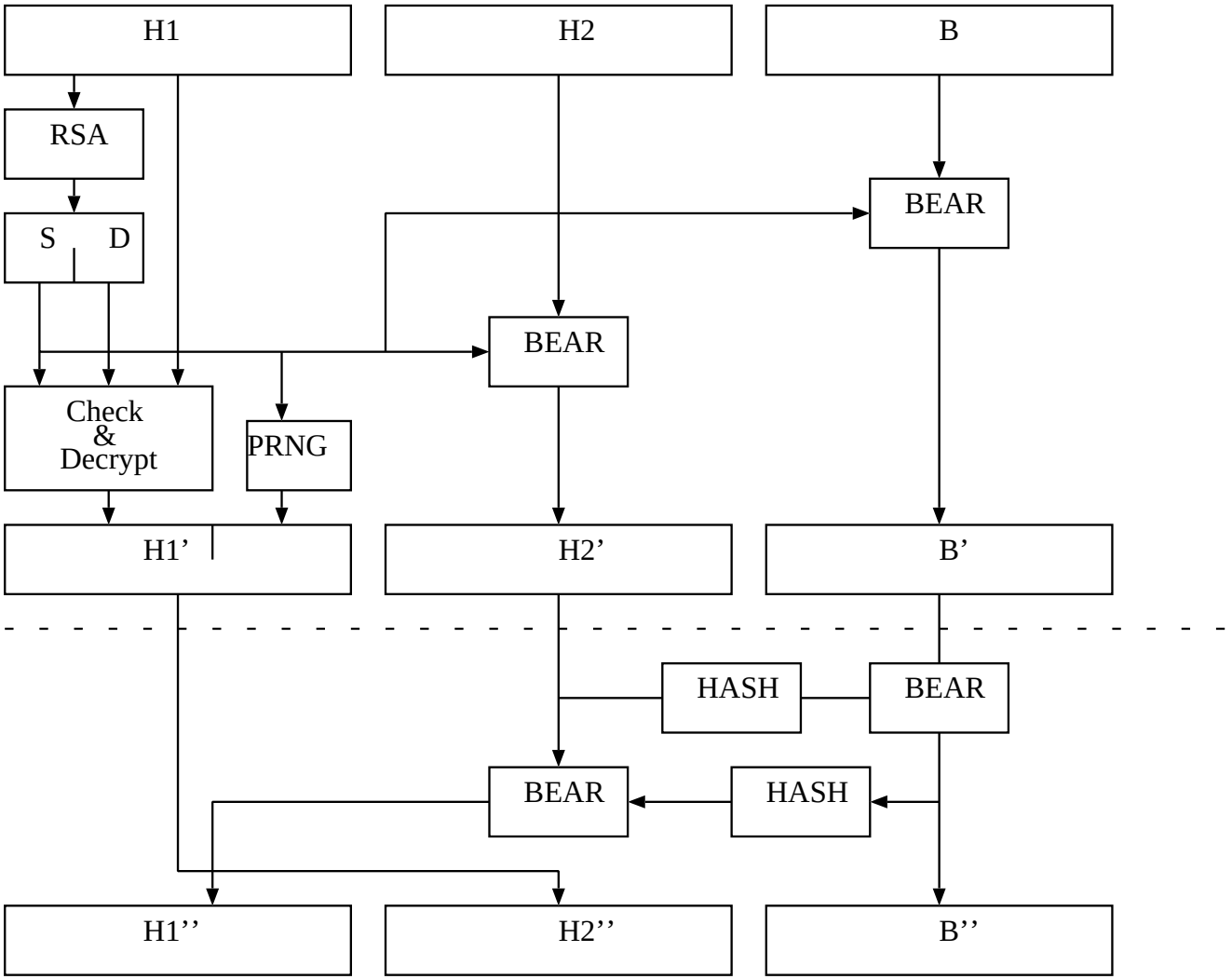
Headers, sub-headers and the overall structure of a mixminion packet.



How to build a mixminion packet

Blackboard. . .

The decoding step



Why is it secure?

(Informal argument – do no lynch!)

- To a passive attacker: The packet always looks indistinguishable from RSA heading followed by random noise.
- To a corrupt mix: No information about route length or position in the route (But first, last and swap). No information about if it is a reply or a normal packet.
- Tagging attacks...
  - Adversary has to tag close to Alice, and only in forward path.
  - If first header is tagged: message is dropped at first honest mix.
  - If second header or body is tagged: Final address and body are lost after first honest mix.
  - Cannot replay a slightly modified onion!

Where to get more information

- Mixminion Specifications (3 documents)
  `http://mixminion.net`
- Mixminion Architecture (IEEE S&P 2003)
- My Ph.D. Thesis (UCAM-CL-TR-594, Chapter 5)
- Beyond: The Minx Packet format (WPES 2004)