¡Cebolla!

spanish("onion") $\longrightarrow$ "cebolla". Pronounce it "say-boy-ya". It has **nothing** to do with ebola.

Pragmatic IP Anonymity

Zach Brown
<zab@zabbo.net>

Thanks to Arjan for the TEX slide templates. Take **that**, MagicPoint.

What is this all about?

Some folks don't like that IP addresses are typically in the clear. The recipient can identify the sender, and everyone in the path knows the source and destination addresses.

Some folks don't want the source's identity divulged:

- A corrupt ISP collecting user profiles
- "carnivore" style large-scale monitoring
- Amazon-style user pattern profiling

Mixnets have tried to restrict full participant knowledge to the sender.

Cebolla applies mixnet properties to provide minimal IP anonymity for the sender.

“Mixnets”? Come again?

Described for messages by Chaum in ’81, “layered envelope” paper mail analogy:

- Many agents in a series make up a route for messages to follow.
- The sender puts instructions and message in a nesting of envelopes.
- Each agent opens their envelope and sends on the revealed envelope.
- Final agent opens their envelope to find a letter for the proper recipient.
- In reverse, agents add envelopes as they send to previous agents.

In the end, the sender knows all agents, each agent only knows its neighbours, and the recipient only knows the final agent.

Specific Security Guarantees

- Only the sender, final agent, and recipient should know the identity of the recipient
- The recipient should not know the identiy of the sender
- The first through second-to-last agents should not see cleartext messages
- Rogue agents should not be able to silently corrupt communications
- Rogue agents are welcome to halt the flow of messages
- Rogue agents must collude to discover identies of recipients or senders

Previous Work 1/2

- NRL starts with Onion Routing
  - No source available, as far as I know
  - Patent 6,266,704 assigned to "The USA via the secretary of the Navy"
- Wei Dai describes PipeNets
  - A fully utilized mesh that reorders frames
  - Not implemented
  - Fully utilized links are expensive, frame reordering hurts things..
- Zero Knowledge productizes their Freedom Network
  - Implementation had some "First Attempt" problems
  - No longer operational

Previous Work 2/2

- Mike Freedman and co. publish Tarzan
  - neato reasearch work
  - slightly different security/efficiency/bleeding edge tradeoffs than Cebolla
- XOR-Trees are a .mil approach
  - Synchronized key generators, masks sender and receiver, can lose K-N nodes
  - **wildly** cool; **very** resource intensive

Architecture Overview

- Many nodes running cebolla form an overlay network
- UDP links maintained between certain nodes
- Nodes flood connectivity information througout the overlay network
- Clients also connect to nodes via UDP links
- Clients iteratively build tunnels that are composed of a path of nodes
- Clients negotiate shared secrets with each node in a path
- Crypto performed by clients and nodes does the heavy lifting
- Exposing tunnels as network devices eases use by using stack functionality

the devil is in the details..

Secret Negotiation

Builds shared secret and negotiates options while allowing for snooping.

- Derived from Photuris, one of the myriad IPsec keying proposals
- Uses 2 round trips, 4 messages exchanged
- All state resides in initiator
- Built around Diffie-Hellman for now
- PPP-style negotiation via "offered" and "accepted" sets of options
- Asymmetric in its protection of the initiator's data
- Uses nutty hashing as first pass DoS prevention

Links

- Simple link header includes sequence numbers, tunnel IDs, “next header” payload type.
- Two link headers, to link header encryption keys: no header MAC.
- Kept alive as long as keep-alives flow.
- Can be torn down by explicit messages from the peer.

Tunnels

- Exist as forwarding instructions and crypto state in a client and in a set of hops.
- Endpoints are presented as networking devices on the client and on the final hop.
- Crypto applied as frames traverse the tunnels; provides cebolla's guarantees.
- Negotiated first over a link, then iteratively over itself until complete.

Data Path Through Tunnels

1. Frames entering the tunnel at the client are encrypted for each hop, in reverse order

2. Encrypted frames are forwarded down the link to the first hop with the proper tunnel ID

3. Receiving node looks up the tunnel ID and decrypts its layer

4. This node then rewrites the tunnel ID for the next hop and forwards resulting packet

5. ... rinse and repeat ...

6. Final hop's decryption reveals a plaintext frame

7. Frame is forwarded out the device

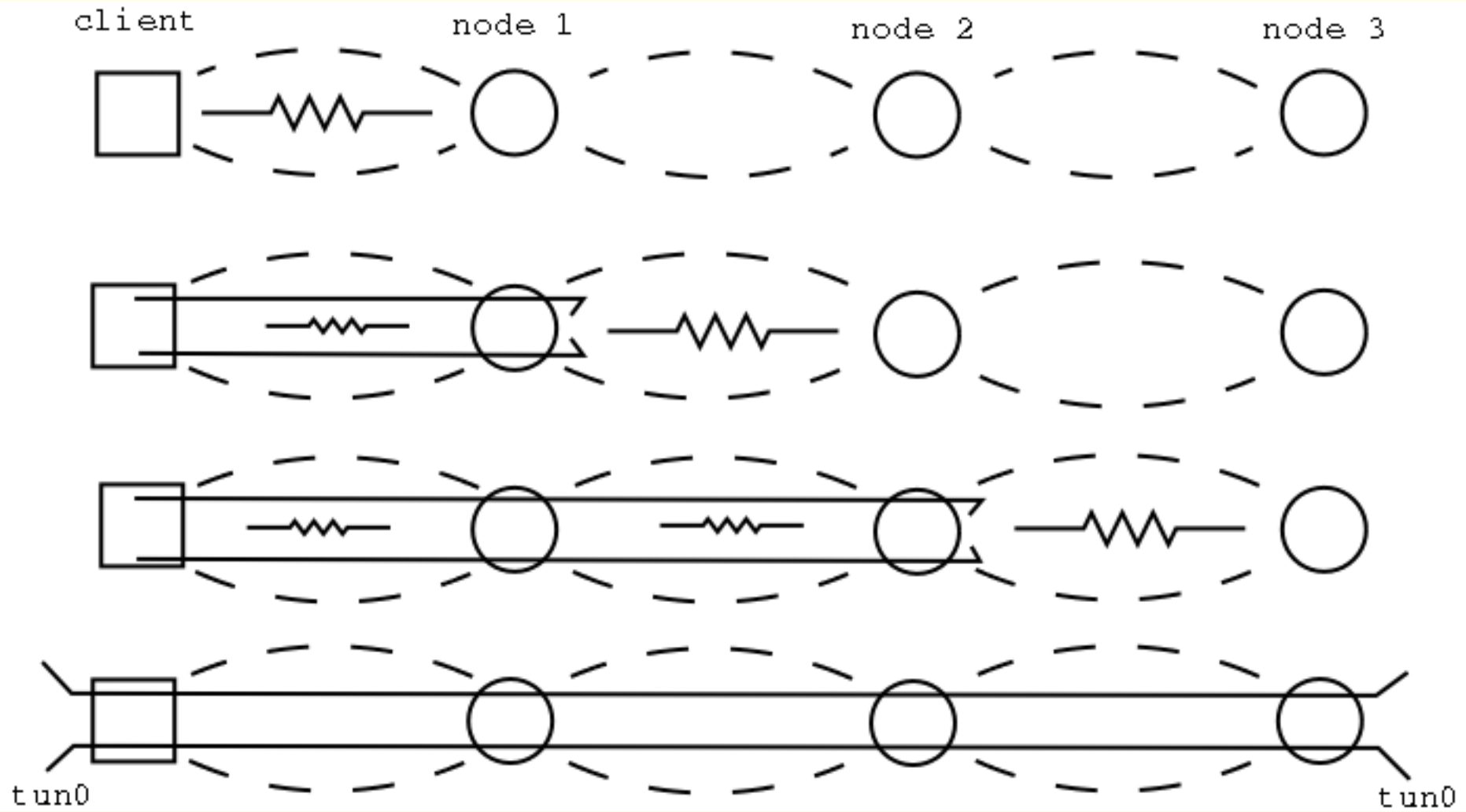
Reverse traffic is encrypted at each hop and multiply decrypted at the client.

Iterative Tunnel Negotiation 1/2

1. First tunnel fragment is negotiated over a link (client to hop 1)
2. Options in the negotiation specify the next hop to use
3. Client negotiates with hop 2 by sending messages down the incomplete tunnel
4. Hop 1 sees messages down the incomplete tunnel and forwards to hop 2
5. When negotiation with hop 2 is complete, hop 1 forwards all tunneled frames to hop 2
6. And so on, for any number of hops, until...
7. Client negotiates an IP address and netmask with the final hop

Tunnel IDs are part of the link header, weird.

Iterative Tunnel Negotiation 2/2



Topology Flooding and Discovery

- Transparently modeled after OSPF
- Nodes regularly send a list of their connected peers to all their peers
- Peers forward to their peers...
- Each transmission is retried until it's acknowledged
- Each node then offers its collection of announcements via rsync
- Clients use synced collections to decide which nodes to connect to
- Announcements could be signed, and do include an enormous sequence number
- Clients can verify that one node's published collection matches others'

Other Bits of Sanity

- IPSec-like sliding-window sequeunce numbers
  - reset at rekeying
  - currently too small
- Spiffy key rotation
  - based on simple messages and decryption feedback
  - pretty simple, seems race-free, concurrent with stream
- Loadable crypto modules
  - OpenSSL Blowfish
  - Diffie-Hellman
  - Enigma (so cool, **so slow**)

Implementation Goals

Goals:

- Portable to not-sucky unices: Linux, FreeBSD (and OS X), Solaris.
- It's an experiment, we want it implemented quickly and it should be agile.
- It should not be wildly complicated, auditing is important.

Choices:

- glib, OpenSSL, GCC and GNU make
- Single threaded daemon, synchronous mainloop
- Deployment requires the user-space daemon, /dev/net/tun, and stack configuration

Further Work

Necessary grunt work:

- Flushing out crypto coverage and authorization
- Security audit (read: convince Chris Evans)

Fun featuritis:

- Use more rigorous "hashcash"-style challenges as DoS prevention
- GUI that manages tunnels with lots of blinkies
- Signed BPF rules in certificates for tunnel exits to apply
- LSAs could contain public information about the node operator, location, resources, operating system, etc...

¿Hay preguntas?