Onion Routing

Varun Pandey

Dept. of Computer Science,

Virginia Tech

What is Onion Routing ?

a distributed overlay network to anonymize TCP based routing

- Circuit based (clients choose the circuit)
- Each node in the path called “ONION” router (OR)
- OR knows only its predecessor or successor but no other node of the circuit
- Traffic flows in fixed-size cells
  - A cell has many layers
  - Each layer “peeled off” by a symmetric key at each node

Dingledon *et. al. ‘s* paper on “TOR: The second generation onion router” provides improvements over the previous designs of onion routing

- First: the design of original Onion Routing implementation as proposed by Goldschlag *et. al.* in their paper on “Hiding Routing Information”.
- aim was to limit vulnerability by **traffic analysis**
- Provide anonymous socket connection by proxy servers
- Only the initiator knew all the nodes. No other node, including the responder need to know about the any other node except the predecessor and successor.

- What is Traffic Analysis?

Analyze traffic patterns instead of data to deduce who is communicating with whom.

In most networks even if data is encrypted, routing information is sent in the clear for router to know the destination of the packet.

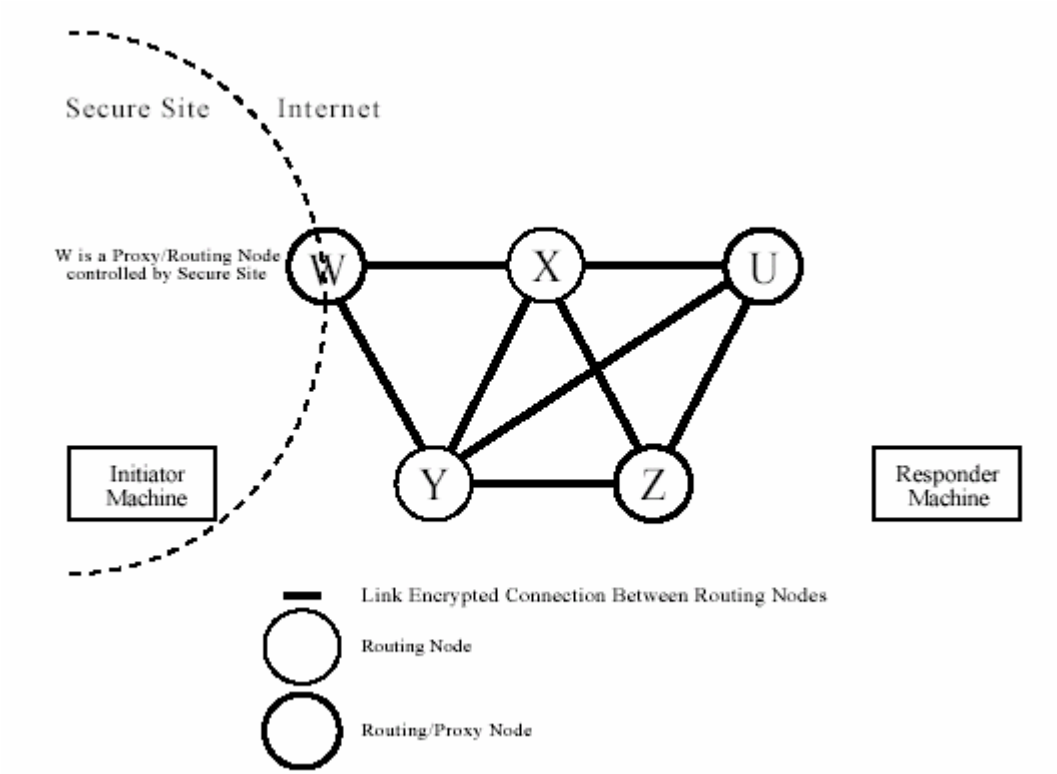
The goal of OR is to prevent giving away identities by traffic analysis on a public network

For e.g. a researcher who used the WWW to collect data from a variety of resources. Although each piece of information that he receives is publicly known, it may be possible for an outside observer to determine his sensitive interests by studying the patterns in his requests. OR makes it very difficult to match his HTTP requests to his site. (from “Hiding Routing Information” by Goldschlag *et.al.*).

OR provide bi-directional communication, without requiring that the responder know the initiator’s identity or location. Individual messages are not logged. Messages can included a “Reply” Onion that permit a later reply to the sender without knowing his address and without keeping the virtual circuit open.

- So the key is the “ONION”.
- How to set up the circuit using “ONIONS”?
- How to use the circuits set up?
- How to destroy the circuits?
- Vulnerabilities?

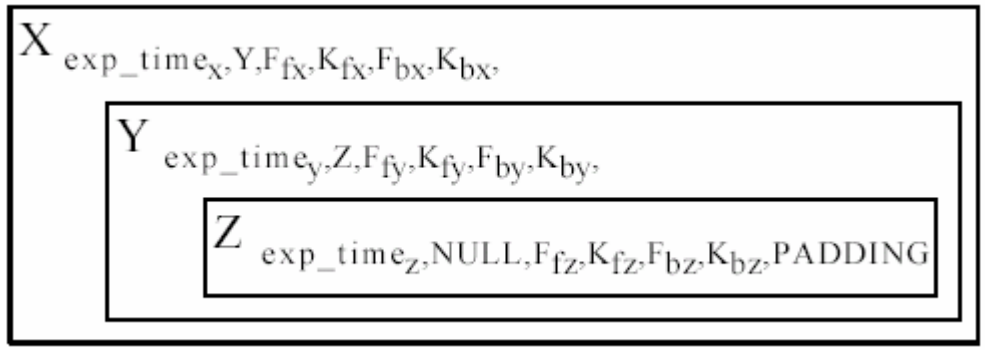
Routing Topology



From “Hiding Routing Information” by Goldschlag et. al. 1996

- ONION: payload with layers over layers of encrypted routing information around it. Guides the construction of VC.
- When an Onion is received each node knows who just sent him him Onion, whom to pass on the onion, but
  - Knows nothing about the other nodes
  - Nor about how many nodes there are in the chain
  - Nor about what is his place in the link (unless he is the last)

A Forward Onion



A Forward Onion

Node PK_x receives:

{exp_time, next_hop, F_f, K_f, F_b, K_b, payload} $_{PKx}$

From “Hiding Routing Information” by Goldschlag et. al. 1996

- Note: for the responder proxy the next_hop field is NULL.
- For any intermediate node, the payload is another onion.
- The exp_time is used to detect replays.
- At each node the ONION shrinks because of peeling off of a layer. This can reveal route info because of diminishing size. To prevent that, random bits = size of the peeled off layer is appended at the end.
- Need to make all onions of the same size to prevent traffic analysis and to hide the length of the chain from the responder's proxy --- for that the initiator's proxy will pad the central payload according to the size of the onion i.e. the no. of hops.

- Creating the circuit

[circuit_identifier, command, data (Onions)]

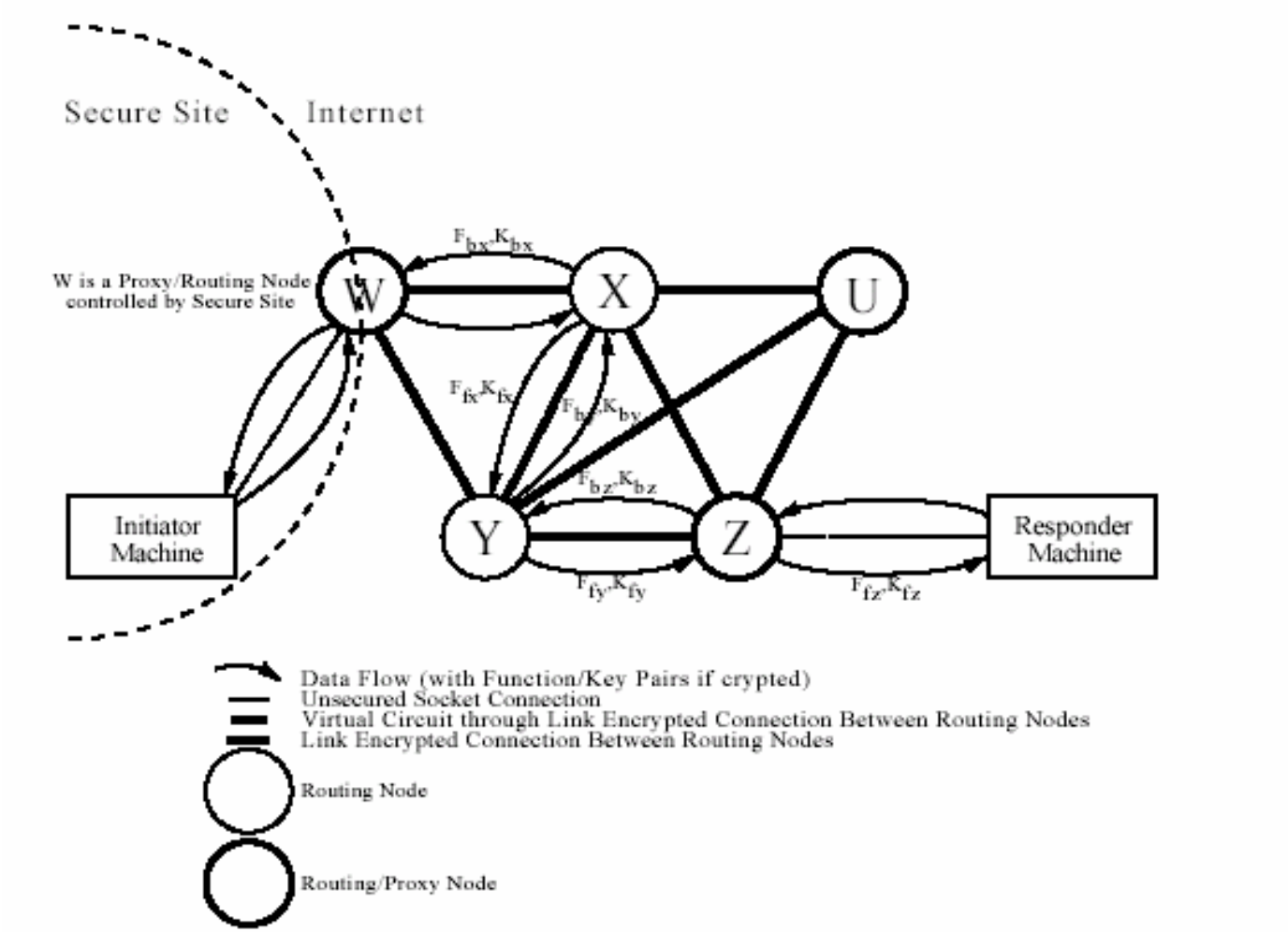
(in link encrypted connection)

command -> create

data

destroy (forwarded in both directions)

A Virtual Circuit



From “Hiding Routing Information” by Goldschlag et. al. 1996

- Loose Routing
- Onion for a loose routing

  $\{exp\_time, next\_hop, max\_loosecount, F_f, K_f, F_b, K_b, payload\}_{PKX}$

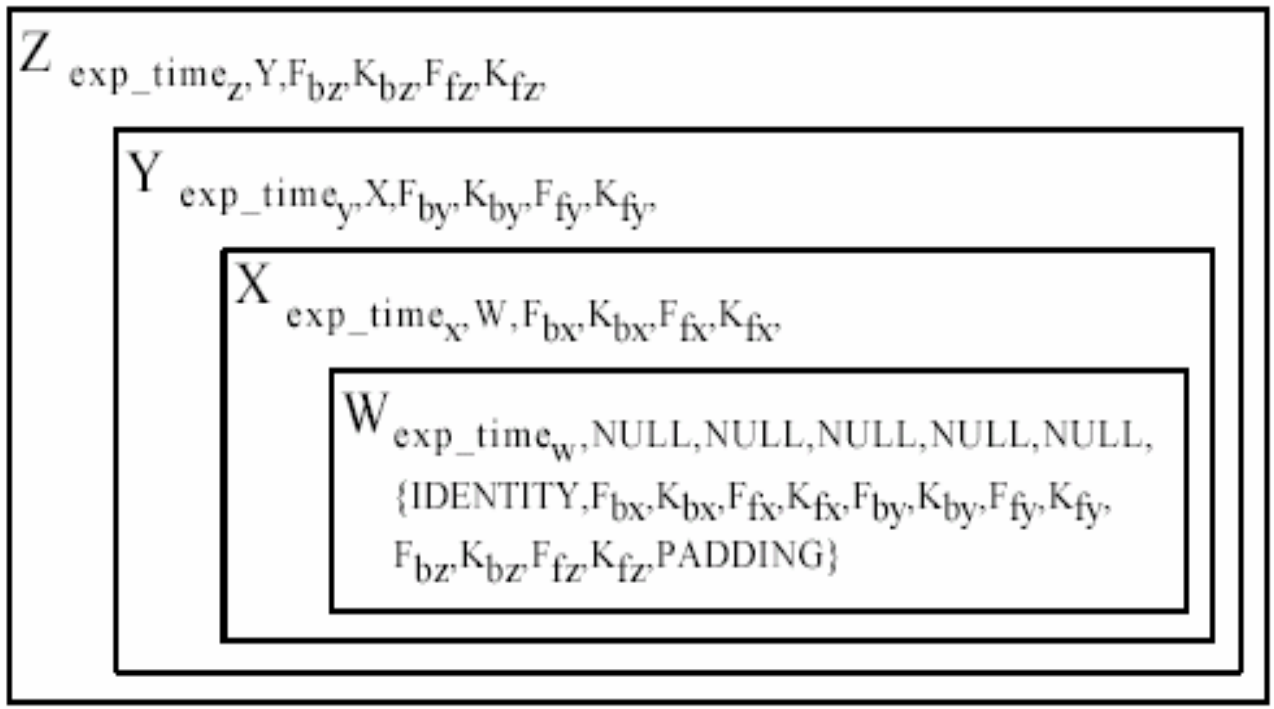
- Reply Onions

  primary difference with Forward Onions

  payload of the forward onion can be effectively empty

  the reply onion payload has enough info to enable the initiator's proxy to reach the initiator and all the Function key pairs of the circuit

Reply Onion



From “Hiding Routing Information” by Goldschlag et. al. 1996

- Improvements over the original Onion Routing as sighted by the paper on TOR by Dingledine *et. al.*
  - Perfect forward secrecy
  - Seperation of “protocol cleaning” from anonymity
  - Many TCP streams can share one circuit
  - Leaky pipe circuit methodology
  - Congestion Control
  - Directory servers
  - Variable exit policies
  - End-to-end integrity checking
  - Rendezvous points and hidden services

Cells in TOR

2	1	509 bytes
CircID	CMD	DATA

2	1	2	6	2	1	498
CircID	Relay	StreamID	Digest	Len	CMD	DATA

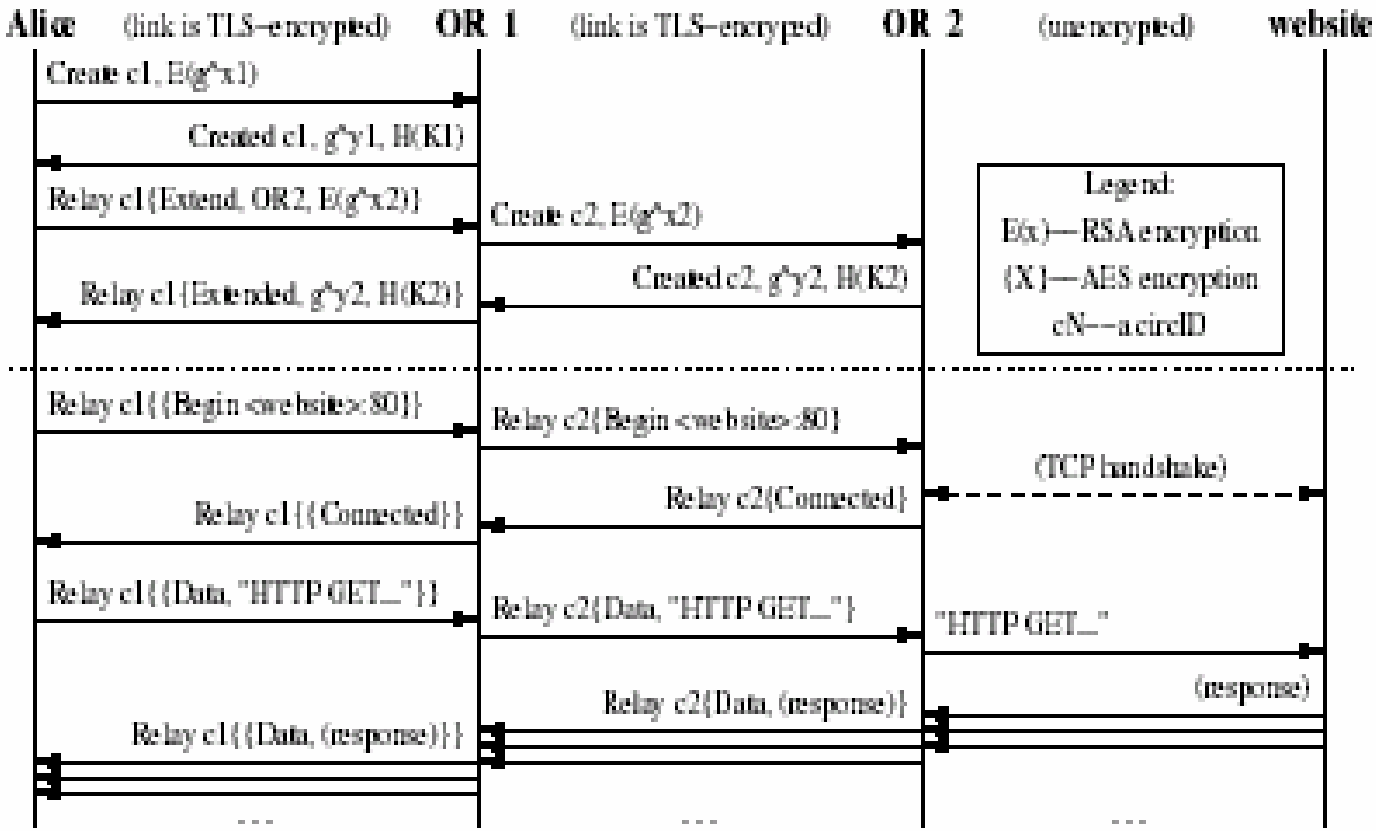
Cells are Controls cells or Relay cells.

Control cells CMDs are: padding, create, destroy

Relay cells CMDs are: relay data, relay begin, relay teardown, relay connected, relay extend, relay truncate, relay sendme, relay drop

NOTE: Each CircID now can have multiple StreamID

Building a two-hop circuit



From "TOR: The Second-Generation Onion Router" by Dingledine et. al.

- Rendezvous points are used to build "*location-hidden*" services.

Goals in building Rendezvous points:

- Access control
- Robustness
- Smear-resistance
- Application transparency

Rendezvous in TOR

- Steps:
  - Bon generates a long-term key pair to identify his service
  - Chooses some introduction points, advertises them (signing them with his public key) on the lookup service
  - Builds a circuit to each of his introduction points
  - Alice learns about Bob's service out of band
  - Alice chooses an OR as her rendezvous point (RP) for her connection to Bob's service, builds a circuit to it, gives it a randomly choosen "rendezvous cookie" to recognize Bob.
  - Alice opens an anonymous stream to one of Bob's introduction point, and gives it a message (encrypted with Bob's public key) telling it about herself and random cookie, her RP and the start of a DH handshake. The introduction point sends a message to BOB
  - Bob builds a circuit to Alice's RP and sends the rendezvous cookie, the second half of the DH handshake, and a hash of the session key they now share.
  - The RP connects Alice's circuit to Bob's. Note that RP can't recognize Alice, Bob, or the data they transmit.
  - Alice sends a relay begin cell along the circuit. It arrives at Bob's OP, which connects to Bob's webserver.
  - An anonymous stream has been established.

Attacks and Defenses in TOR

- Passive attacks
  - Observing user traffic patterns
  - Observing user content
  - Option distinguishability
  - End-to-end timing correlation
  - End-to-end size correlation
  - Website fingerprinting

- Active Attacks
  - Compromise keys
  - Iterated compromise
  - Run a recipient
  - Run an onion proxy
  - DoS non-observed nodes
  - Run a hostile OR
  - Introduce timing into messages
  - Tagging attacks
  - Replace contents of unauthenticated protocols
  - Replay attacks
  - Smear attacks
  - Distribute hostile code

- Attacks against rendezvous points
  - Make many introduction requests
  - Attack an introduction point
  - Compromise an introduction point
  - Compromise a rendezvous point

- Thanks for your attention.