TOR : THE SECOND GENERATION
ONION ROUTER

Özlem Kahraman ER
What is Onion Routing?

- Creates a random route from source to destination
- Each router is only aware of it's adjacent hops
- The route through the “onion field” is determined by the client
- Data is encrypted, including next and previous hop info (header)
What is Tor?

• Second Generation Onion Routing Network
• Provides a client / proxy for interfacing with Onion Routers
• Speaks SOCKS to the local operating system
• Speaks TLS to the Onions Routers
Design Goals and Assumptions-1

• Goals

  – **Deployability** – conformant for real word use
  – **Usability** – more usable, more users, more anonymity, no platform change needed
  – **Flexibility** – is a base for future designing
  – **Simple designing**

*Main goal is to frustrate attackers from linking communicating partners.*
Design Goals and Assumptions-2

• Non Goals
  – Not peer-to-peer
    • Thousands of short lived servers, many controlled by adversary
  – Not secure against end-to-end attack
    • Connection between OP and entry node is the weak point
The TOR Design 1
The Tor Design (cells and used keys)

- **Cells**
  - 512 byte cells
  - Command types: control, relay
    - control cell types: padding, create-created, destroy
    - Relay cell types: relay begin, relay connected, relay extend, relay extended, relay data, relay end, relay truncate, relay
  - 1- **create cell** to construct circuit
  - 2- **relay cell**
  - Sign digest + (header-payload) encrypted with shared Diffie Hellman key

Onion routers (OR)
- Maintains TLS connection with each node
  - long - term identity key
    - Router discription, directoryes.
  - Short – term
    - Onion key (the private key for Public key cryptography) in TLS.
    - Shared secret key with other ORs (Diffie Hellman handshake) shared by TLS.
The Tor Design (constructing a circuit)

Alice (link is TLS-encrypted)  OR 1 (link is TLS-encrypted)  OR 2 (unencrypted)  website

Create c1, E(g^x1)

Created c1, g^y1, H(K1)

Relay c1{Extend, OR2, E(g^x2)}

Created c2, g^y2, H(K2)

Relay c1{Extended, g^y2, H(K2)}

Create c2, E(g^x2)

Relay c1{Begin <website>:80}

Relay c2{Begin <website>:80}

Relay c1{{Connected}}

Relay c2{Connected}

Relay c1{{Data, "HTTP GET..."}}

Relay c2{Data, "HTTP GET..."}

Relay c1{{Data, (response)}}

Relay c2{Data, (response)}

Legend:
E(x)—RSA encryption
{X}—AES encryption
cN—a circID

(TCP handshake)

"HTTP GET..."

(response)
Tor Features

• Congestion Control
• Directory Servers
• Integrity Checking
• Configurable Exit Policies
• Perfect forward Secrecy
• Location-Hidden services, “Rendezvous Points”
Congestion Control

• Enough user choose the same OR1-OR2 connection for their circuits.
  – Methods:
    1- circuit level throttling
      * OR keeps two window:
        packaging window, delivery window
      *if packaging window = 0 then wait relay sendme cell
    2-stream level throttling
      packaging window, delivery window
      *if pending bytes > 10 send relay sendme cell, not after every enough data.
Directory Servers

• In Original Onion Routing each router floods its state to network periodically.
  – Because of delays Directory Servers are not syncron at a time. This helps attacker
• TOR uses trustworthy routes as directory servers.
  – DS signs the directory, OR sends signed statement and download the directory periodically.
  – OR who has invalid key are not in directory
• Variety of attacks remain
  – Attacker can control DS.
  – Gives only nodes he controls,
  – Differences between DS.
Directory Servers Assumptions

• All participants agree on the set of Directory Servers
• Needs a threshold consensus of the current state of the network.
• When a consensus directory cannot be reached then human administration is needed.
Integrity Checking on streams

• Any integrity checking in Original Onion Routing
• TOR uses TLS, public key - private key cryptography together and attacker can’t modify data.
• Integrity is checked at the edges. Only exit node can control the digest.
Location-Hidden services, “Rendezvous Points”

1. Server Bob creates onion routes to Introduction Points (IP)
2. Bob gets Service Descriptor incl. Intro Pt. addresses to Alice
   - In this example gives them to Service Lookup Server
**Location-Hidden services, “Rendezvous Points”**

2'. Alice obtains Service Descriptor (including Intro Pt. address) at Lookup Server
Location-Hidden services, “Rendezvous Points”

3. Client Alice creates onion route to Rendezvous Point (RP)
4. Alice sends RP addr. and any authorization through IP to Bob
Location-Hidden services, “Rendezvous Points”

5. If Bob chooses to talk to Alice, connects to Rendezvous Point
6. Rendezvous point mates the circuits from Alice and Bob
Attacks on TOR

- Traffic analysis attacks
- Compromise keys (perfect secrecy)
- Run on onion proxy
- Replace contents of unauthenticated protocols
  - Don’t use HTTP
- Run a hostile OR
  - Make itself trustworthy to a Directory Server
- Destroy directory servers
- Make many interaction nodes as a Rendezvous Point
  - Defence: Filtering in Introduction Points
- Disrupt an introduction point
  - New introduction point will be published and Introduction points published only for trustworthy clients.
- Compromise an introduction point
  - Flood interaction requests to bob
  - Bob recognise a flood and close the related circuit.
Open questions

• What would be period of refreshing the circuits.
• What would be the hop count in a circuit.
• Is random path length is neccessary.
• Hydra topology could be used.
  – Many inputs and few exit nodes.
Future Directions

• Bandwidth
  – ORs have good bandwidth and latency,
  – ORs can advertise their bandwidth and selecting nodes could be done according to this info.

• Incentives (teşvik)
  – Reward users with better anonymity, more nodes means more anonymity.

• Better directory distribution
  – Entire network state downloaded every 15 minutes. Only updates could be downloaded.

• Caching at exit nodes
  – Exit nodes should run a caching proxy – forward secrecy is weakened

• Wider-scale deployment
  – Having more users, evaluation of design principles will be more realistic (robustness – latency tradeoff, abuse prevention)