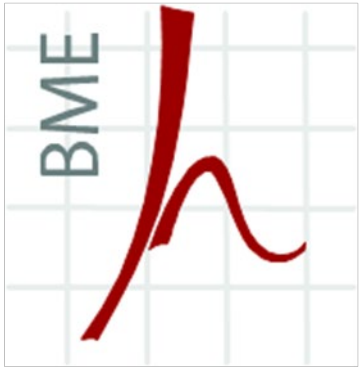


Anonymization networks, Tor

Dr. Boldizsár Bencsáth
assistant professor
BME Híradástechnikai Tanszék
Lab of Cryptography and System Security (CrySyS)
bencsath@crysys.hu

Introduction

- privacy is the right of an individual to control how information about him/her is collected, stored, and shared
  - people don't like if their personal information such as their religion, sexual orientation, political affiliations, or personal activities are revealed
  - this may be to avoid discrimination, personal embarrassment, or damage to one's professional reputation

- new technologies also create new ways to gather and process private information

- today, as large scale information systems become more common, there is a lot of information stored in many databases worldwide and an individual has no way of controlling (or even knowing) of the information about themselves that others may have access to
  - such information could potentially be sold to others for profit and/or be used for purposes not known to the individual of which the information is about
  - consequences of a violation of privacy can be more severe

Introduction (cont'd)

- privacy is not really a technical issue: it needs legislation (laws) and law enforcement

- however, technical solutions can help to retain control over private information in some cases
  - encryption and access control techniques
  - anonymous communication techniques
  - in general: privacy enhancing technologies

Objectives of this lecture

- see some examples on how technical approaches can help to retain privacy in IT systems

- get an introduction into anonymous communication techniques used over the Internet
  - understand how a MIX network works
  - get some more insight into the engineering details of a practical anonymity system (Tor)

Basic anonymity concepts

- What do we want to hide?
  - sender anonymity
    - attacker cannot determine who the sender of a particular message is
  - receiver anonymity
    - attacker cannot determine who the intended receiver of a particular message is
  - unlinkability
    - attacker may determine senders and receivers but not the associations between them (attacker doesn't know who communicates with whom)

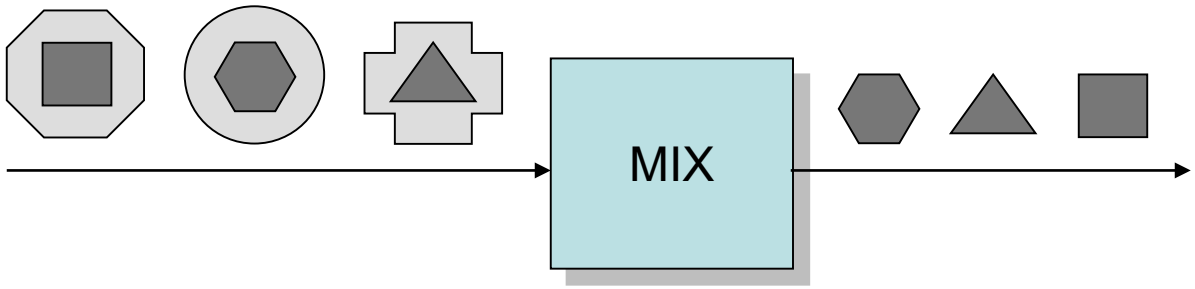
- From whom do we want to hide this?
  - external attackers
    - local eavesdropper (sniffing on a particular link (e.g., LAN))
    - global eavesdropper (observing traffic in the whole network)
  - internal attackers
    - (colluding) compromised elements of the anonymity system
  - communication partner

Anonymizing proxy

- application level proxy that relays messages back and forth between a user and a service provider
- properties:
  - ensures only sender anonymity with respect to the communicating partner (service provider does not know who the real user is)
  - a local eavesdropper near the proxy and a global eavesdropper can see both the sender and the receiver information
  - proxy needs to be trusted for not leaking information (it may be coerced by law enforcement agencies!)
  - even if the communication between the user and the proxy, as well as between the proxy and the server is encrypted, a naïve implementation would have the same properties (weaknesses)

MIXes

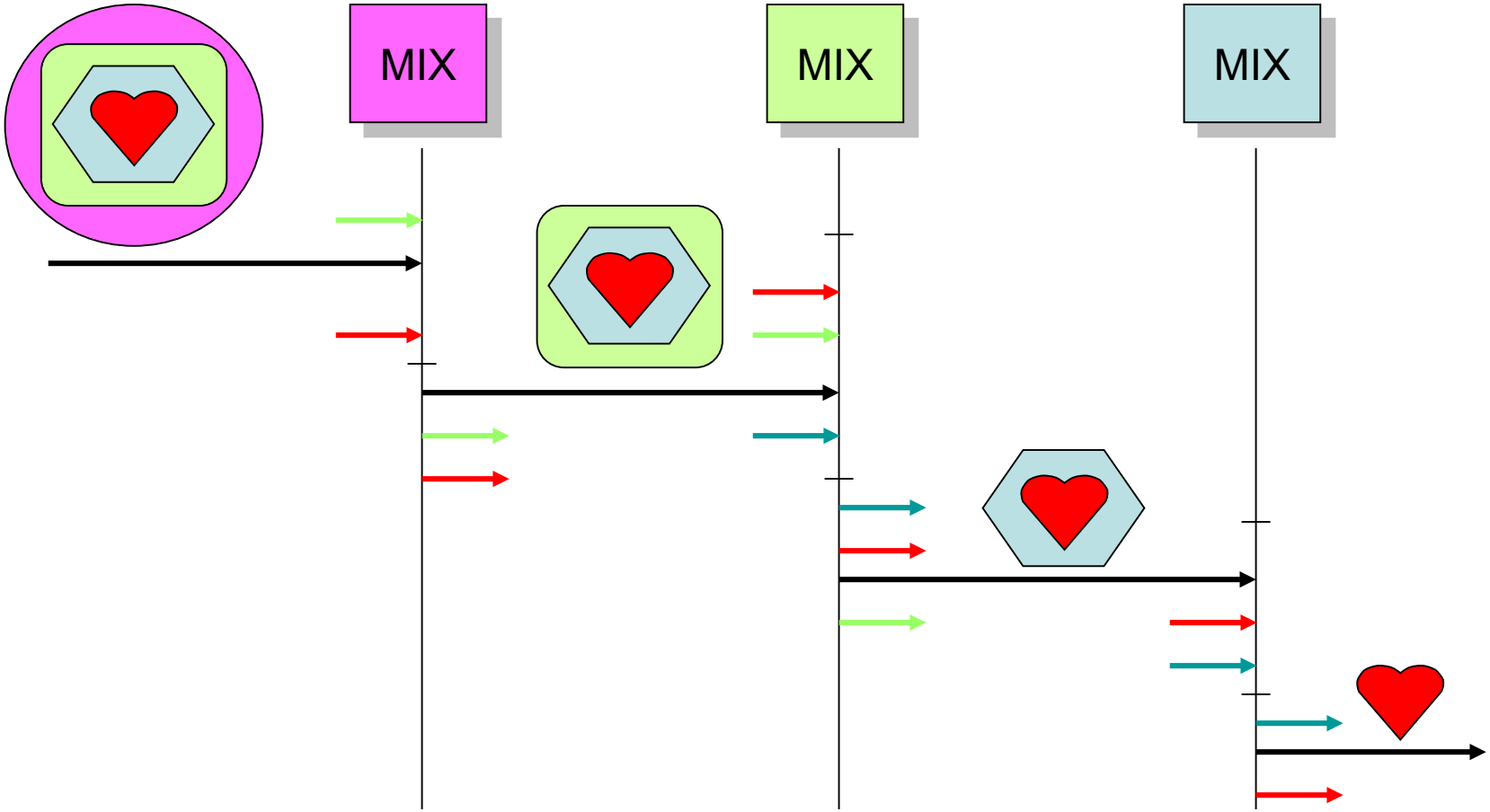
- a MIX is a proxy that relays messages between communicating partners such that it
  - changes encoding of messages
  - batches incoming messages before outputting them
  - changes order of messages when outputting them
  - (may output dummy messages)



- properties:
  - sender anonymity w.r.t. communication partner
  - unlinkability w.r.t. global (and hence local) eavesdroppers
  - the MIX still needs to be trusted

MIX cascade (network)

- defense against colluding compromised MIXes
  - if a single MIX behaves correctly, unlinkability is still achieved



Implementation of MIXes

- each MIX has an RSA key pair
- sender of a message m selects a path through the MIX network and encodes the message iteratively for each MIX on the path

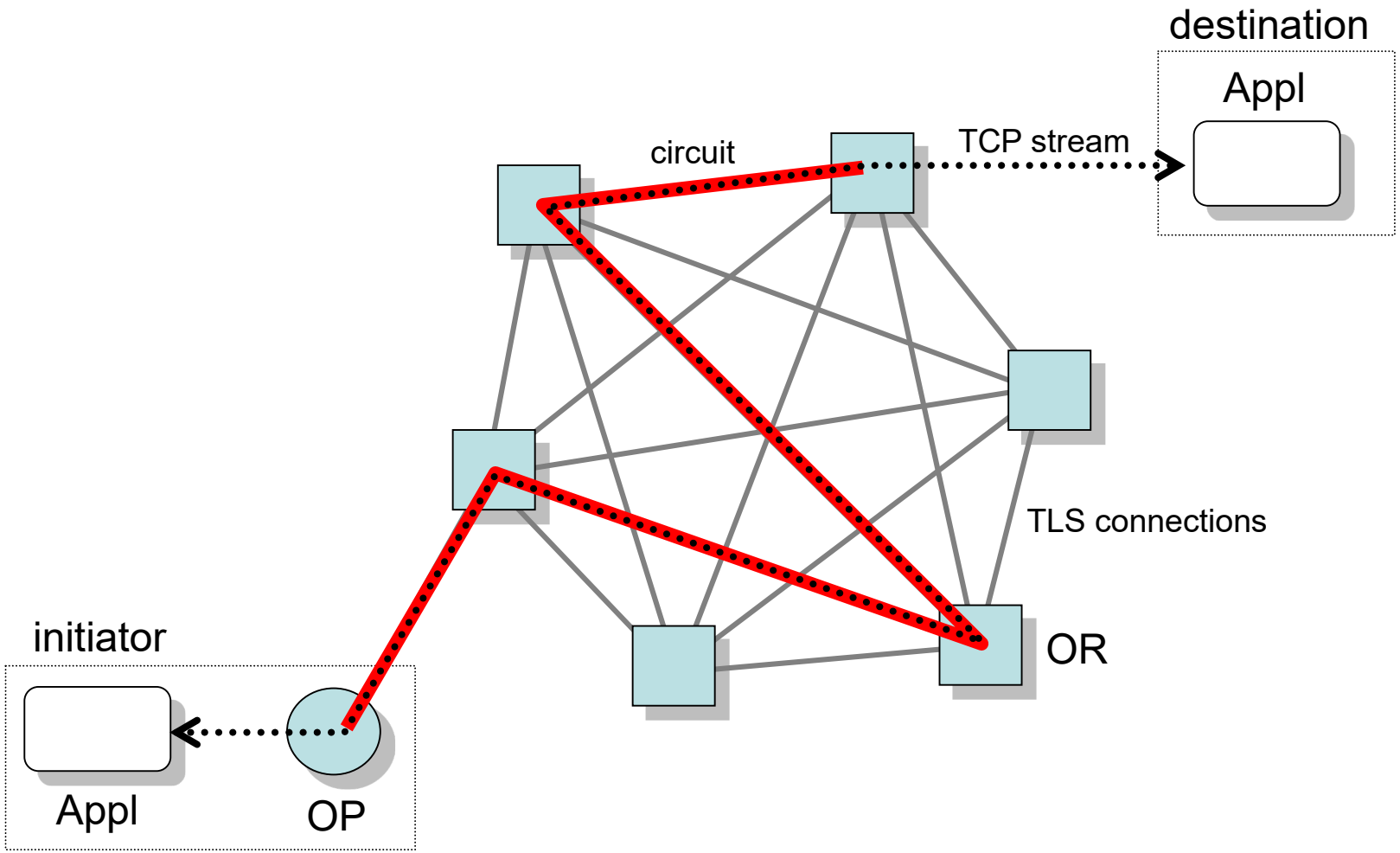
$$MIX_1 \mid E_{MIX1}(MIX_2 \mid E_{MIX2}(MIX_3 \mid \ldots E_{MIXn}(m \mid rnd_n)\ldots \mid rnd_2) \mid rnd_1)$$

  where $E_{MIXi}(.)$ is RSA encryption with the public key of MIX_i and rnd_i is some sufficiently large random number (salt)
  - alternatively, a probabilistic encryption scheme (e.g., ElGamal or RSA with PKCS#1 formatting) could be used
- each MIX decodes the message with its private key and forwards it to the next MIX indicated in the decoded message
- message m may contain a “return address”
  - an iteratively encrypted structure, where layer i is encrypted with the public key of the i-th MIX on the return path and contains
    - the identifier of the next MIX on the return path
    - a secret key to be used for encrypting the content of the reply
    - layer i-1 of the return address

The Tor network

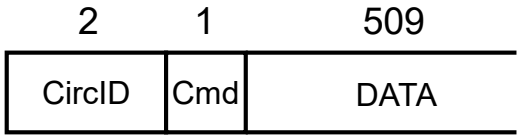
- Tor is a low-latency (real-time) mix-based anonymous communication service
- the Tor network is an overlay network consisting of onion routers (OR)
- ORs are user-level processes operated by volunteers in the Internet
  - a few special directory servers keep track of the ORs in the network
  - each OR has a descriptor (keys, address, bandwidth, exit policy, etc.)
  - each OR maintains a TLS connection to all other ORs
- users run an onion proxy (OP) locally, which establishes virtual circuits across the Tor network, and multiplexes TCP streams coming from applications over those virtual circuits
- the last OR in a circuit connects to the requested destination and behaves as if it was the originator of the traffic

The Tor network illustrated

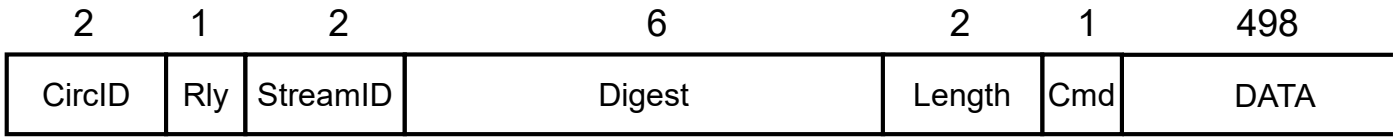


Cells

- data within the Tor network are carried in fixed sized cells (512 bytes)
- cell types
  - control cells
    - used to manage (set up and destroy) circuits



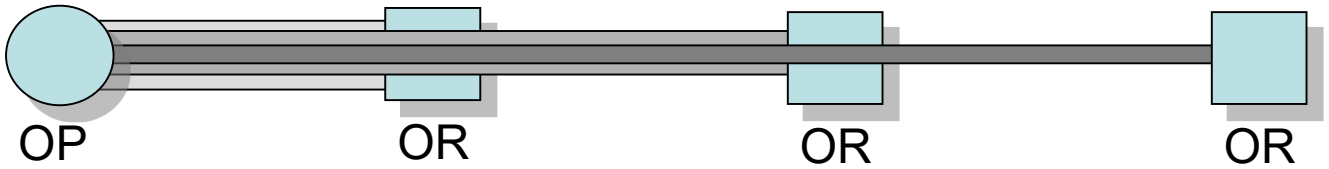
  - relay cells
    - used to manage (extend and truncate) circuits, to manage (open and close) streams, and to carry end-to-end stream data



Setting up a circuit

- circuits are shared by multiple TCP streams
- they are established in the background
  - OPs can recover from failed circuit creation attempts without harming user experience
- OPs rotate to a new circuit once a minute

- a circuit is established incrementally, in a “telescoping” manner
  - a circuit is established to the first OR on the selected path by setting up a shared key between the OP and that OR
  - this circuit is extended to the next OR by setting up a shared key with that OR; this already uses the circuit established in the previous step
  - and so on…



Establishment of shared keys

- Diffie-Hellman based protocol:

  OP → OR: $E_{PK\_OR}(g^x)$
  OR → OP: g^y | H(K | “handshake”)

  where K is the established key g^{xy}

- properties:
  - unilateral entity authentication (OP knows that it is talking to OR, but not vice versa)
  - unilateral key authentication (OP knows that only OR knows the key)
  - key freshness (due to the fresh DH contributions of the parties)
  - perfect forward secrecy
    - (assuming that OR deletes the shared key K when it is no longer used)
    - if OR is later compromised, it cannot be used to decrypt old (recorded) traffic

Relaying cells on circuits

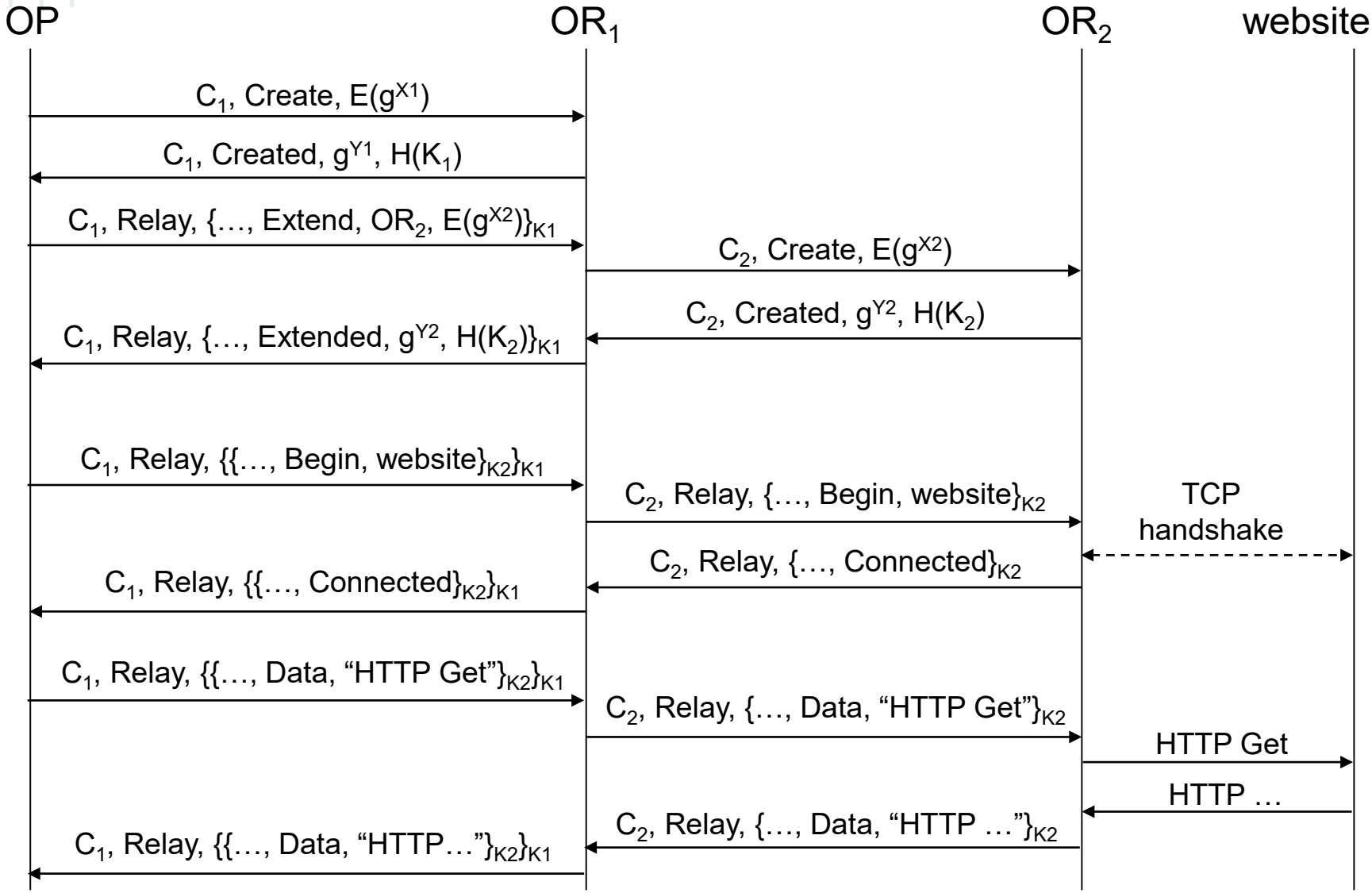
- application data is sent in relay cells
- OP encrypts the cell iteratively with all the keys that it shares with the ORs on the path (onion-like layered encryption)
- each OR peals off one layer of encryption
- last OR sends cleartext data to the destination
- on the way back, each OR encrypts the cell (adds one layer), and the OP removes all encryptions
- AES is used in CTR mode (stream cipher) → encryption does not change the length

Opening and closing streams

- opening:
  - the TCP connection request from the application is re-directed to the local OP (via SOCKS)
  - OP chooses an open circuit (the newest one), and an appropriate OR to be the exit node (usually the last OR, but maybe another due to exit policy conflicts)
  - OP opens the stream by sending a “relay begin” cell to the exit OR
  - the exit OR connects to the given destination host, and responds with a “relay connected” cell
  - the OP informs the application (via SOCKS) that it is now ready to accept the TCP stream
  - OP receives the TCP stream, packages it into “relay data” cells, and sends those cells through the circuit

- closing:
  - OP or exit OR sends a “relay end” cell to the other party, which responds with its own “relay end” cell

Operation illustrated



Exit policies

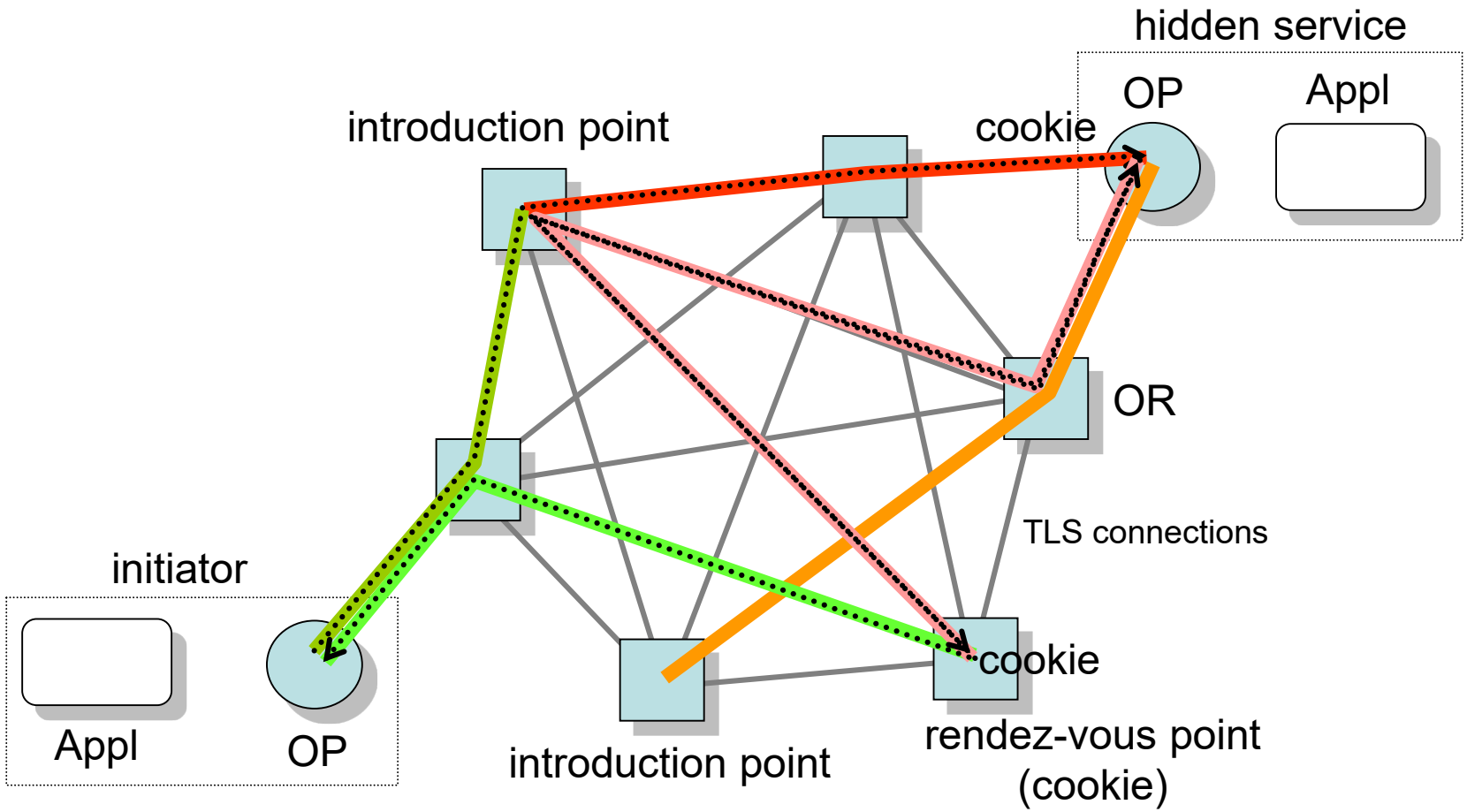
- problem: hackers can launch their attacks via the Tor network
  - no easy way to identify the real origin of the attacks
  - exit nodes can be accused
  - this can discourage volunteers to participate in the Tor network
  - fewer ORs means lower level of anonymity

- solution: each OR has an exit policy
  - specifies to which external addresses and ports the node will connect
  - examples:
    - open exit – such nodes will connect anywhere
    - middleman – such nodes only relay traffic to other Tor nodes
    - private exit – only connect to the local host or network
    - restricted exit – prevent access to certain abuse-prone addresses and services (e.g., SMTP)

Rendez-vous and hidden services

- Tor allows someone to offer a TCP-based service anonymously (without revealing his IP address to the world)
- the server's OP chooses some ORs as introduction points and builds a circuit to each of these introduction points
- the service provider advertises his service and the related introduction points by some anonymous directory service
- the client's OP chooses an OR as the rendez-vous point, builds a circuit to the rendez-vous point, and gives it a random rendez-vous cookie
- the client OP builds a circuit to one of the service introduction points, opens an anonymous stream to the server, and sends the cookie and a reference to the rendez-vous point
- the server OP builds a circuit to the given rendez-vous point and sends the cookie
- the rendez-vous point verifies the cookie and connects the client circuit to the server circuit
- the client establishes an anonymous stream through the new circuit

Rendez-vous illustrated



Some attacks

- end-to-end timing (or size) correlation
  - an attacker watching traffic patterns at the initiator and the responder will be able to confirm the correspondence with high probability
  - it was not the goal of Tor to prevent this

- website fingerprinting
  - an attacker can build up a database containing file sizes and access patterns for targeted websites
  - he can later confirm a user's connection to a given website by observing the traffic at the user's side and consulting the database
  - in case of Tor, granularity of fingerprinting is limited by the cell size

- tagging attacks
  - an attacker can "tag" a cell by altering it, and observing where the garbled content comes out of the network
  - integrity protection of cells prevent this

What is Tor good for?

- anonymous web browsing
- anonymous e-mail
- anonymous remote electronic voting
- …
- carrying out malicious activity anonymously
  - we installed a Tor OR and ran it for 72 hours
  - we could observe 2216 sessions in clear (where our OR was the exit router) where a password was transmitted (most probably on-line password guessing attempts)
  - some statistics:
    - allotracker.com: 1285
    - yahoo.com 59
    - games.amil.ru 18
    - tracker.zamunda.net 140

    - nntp: 25
    - http: 1900
    - pop3: 286
    - other: 5