Computer Security

15. Tor & Anonymous Connectivity

Paul Krzyzanowski

Rutgers University

Spring 2017

Private Browsing

- Browsers offer a "private" browsing modes
  - Apple *Private Browsing,* Mozilla *Private Browsing,* Google Chrome *Incognito Mode,* Microsoft *InPrivate* browsing
- What does it do
  - Do not send stored cookies
  - Do not allow servers to set cookies
  - Do not use or save auto-fill information
  - List of downloaded content
  - At the end of a session
    - Discard cached pages
    - Discard browsing & search history
- Does not protect the user from viruses, phishing, or security attacks

Is private browsing private?

- It doesn't leave too many breadcrumbs on your device
- It limits the ability of an attacker to use cookies
- But
  - Your system may be logging outbound IP addresses
  - Proxies know what you did … so do firewalls & routers
  - Your ISP knows who you are and where you went
  - Web servers get your IP address

Answer: *not really*

Goal

Communicate while preserving privacy

Why?

- Avoid consequences (social, political, legal)
  - E.g., political dissidents
- Avoid geolocation-based services
- Hide corporate activity (who's talking to whom)
- Perform private investigations
- Hide personal info, like searching about diseases you have
- …

Tor & The Tor Browser

- Tor = The Onion Router
- Tor Browser = preconfigured web browser that uses Tor
  - Provide anonymous browsing
- Hosted on a collection of relays around the world
  - Run by non-profits, universities, individuals
- 100K to millions of users
  - Exact data unknown – it's anonymous
  - Terabytes of data routed each second

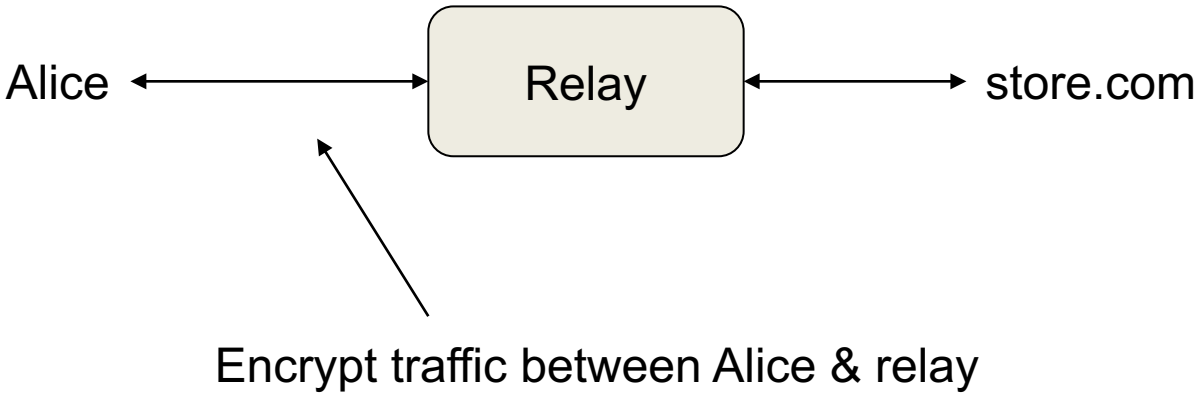
History

- Onion routing developed in the mid 1990s at the U.S. Naval Research Laboratory to protect U.S. intelligence communications
- Additional work by the Defense Advanced Research Projects Agency (DARPA)
- Patented by the U.S. Navy in 1998
  - Naval Research Laboratory released to code for Tor under a free license
- The Tor Project
  - Founded in 2006 as a non-profit organization with support of the EFF

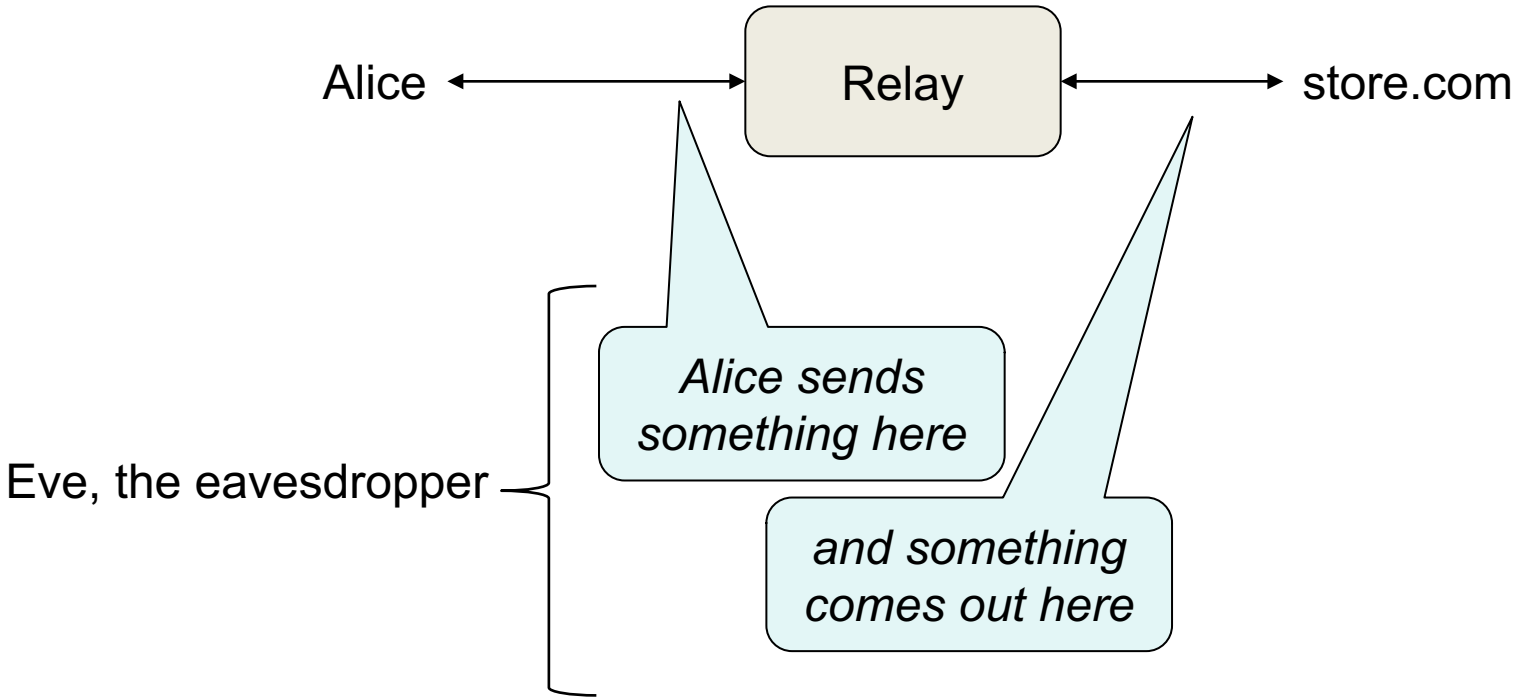
What is anonymity?

- Unobservability
  - Inability of an observer to leak participants to actions
- Unlinkability
  - Inability to associate an observer with a profile of actions
  - *E.g., Alice posts a blog under an assumed name*
    Unlinkability = inability to link Alice to a specific profile

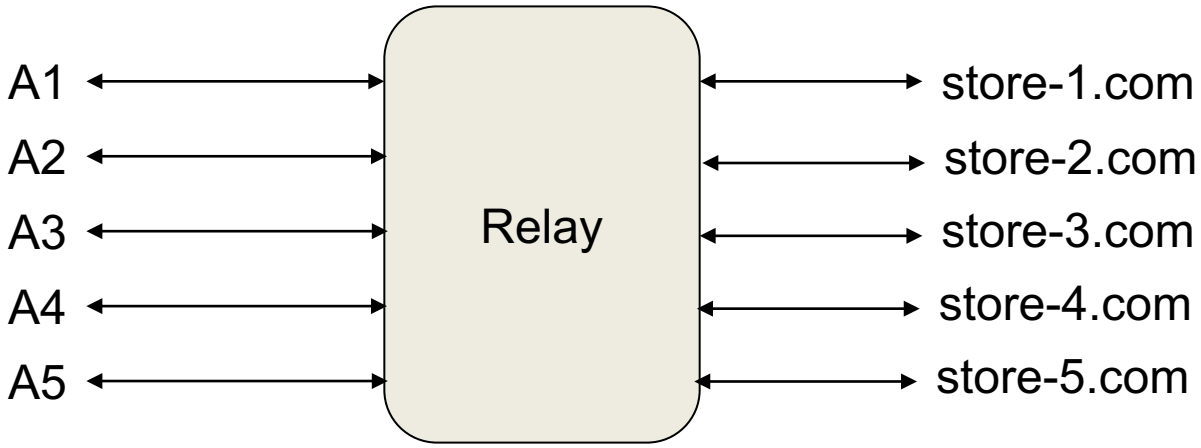
Relay



Relay


Alice
Relay
store.com
Alice sends something here
and something comes out here
Eve, the eavesdropper

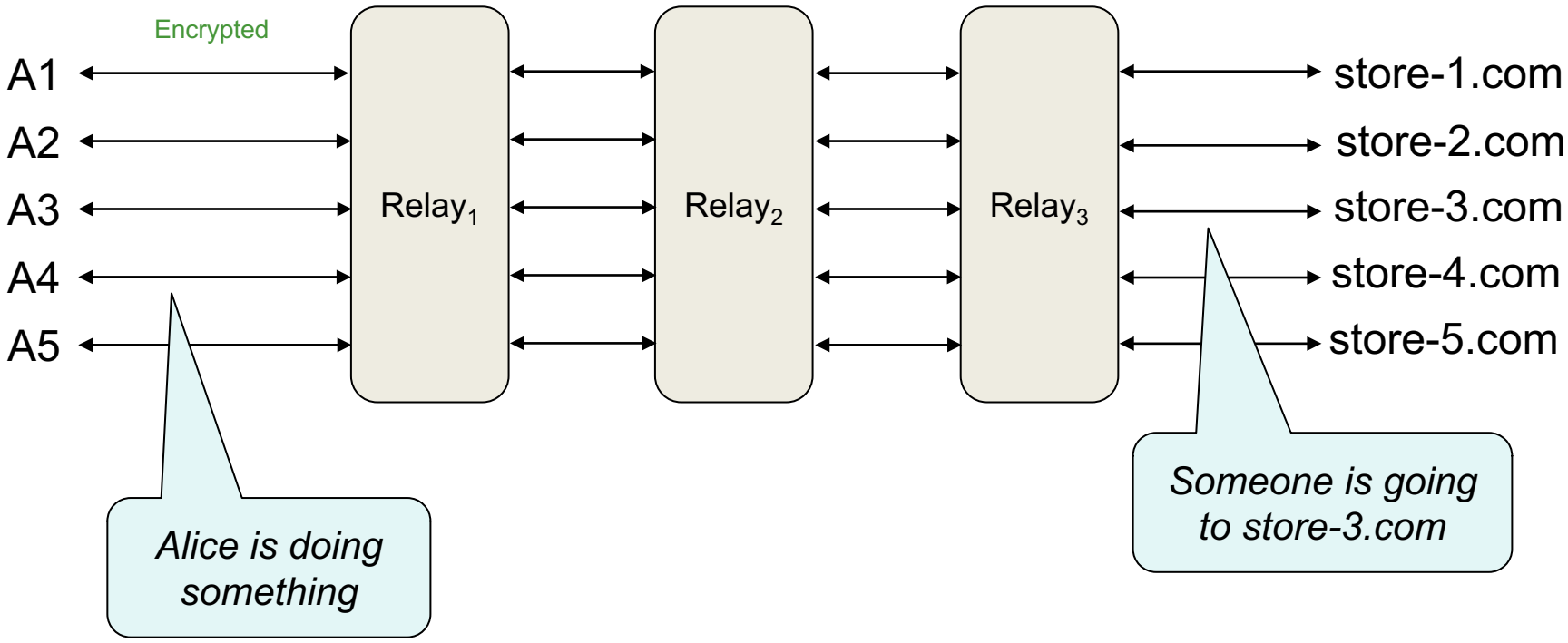
Relay with multiple parties



We can use encrypted connections (TLS) to hide network traffic

What if someone eavesdrops on the relay?

Multiple relays



You cannot see all activity at one relay

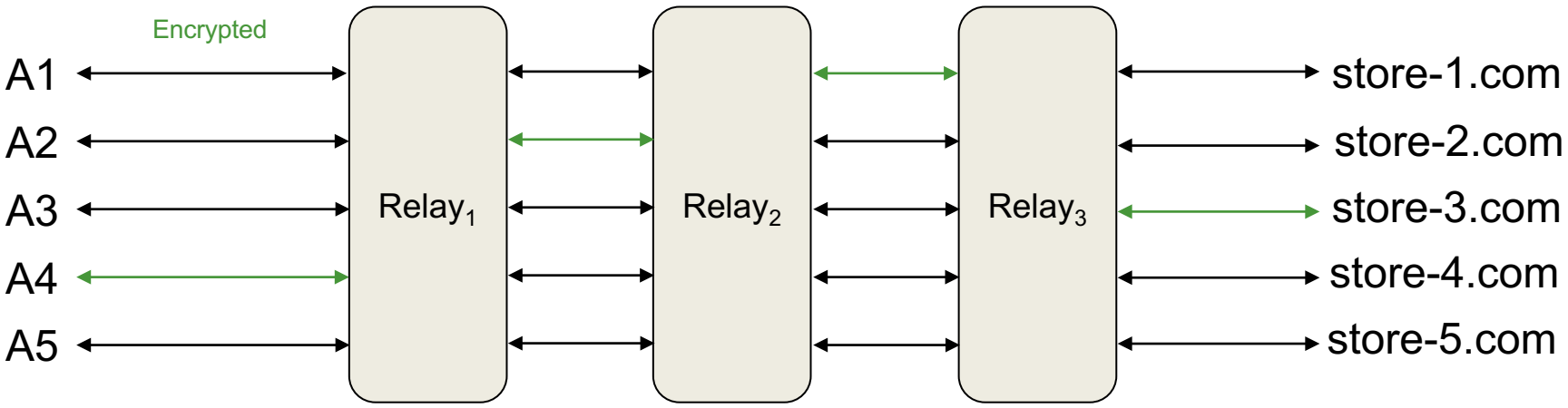
Correlation Attack

- If an eavesdropper watches entry & exit of data
  - She can correlate timing & size of data at the 1st relay with outputs of the last relays
  - If Alice sends a 2 KB request to $Relay_1$ at 19:12:15
    and $Relay_3$ sends a 2 KB request to store-3.com at 19:12:16
    and store-3.com sends a 150 KB response to $Relay_3$ at 19:12:17
    and Alice receives a 150 KB response at 19:12:18
    … *we're pretty sure Alice is talking to store-3.com*

Correlation Attack

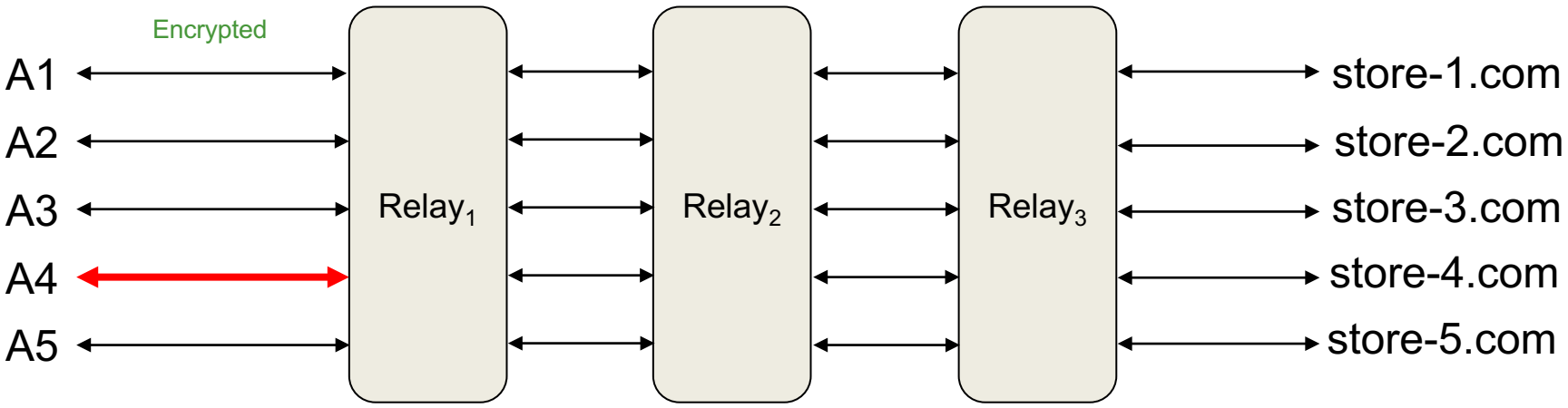
- You can make a correlation attack attack difficult
  - Pad or fragment messages to be the same size
  - Queue up multiple messages, shuffle them, and transmit them at once
- This works in theory but is a pain in practice
  - Extra latency, traffic
  - You still need A LOT of users to ensure anonymity
- Relays should be hosted by third parties to get many different groups as input
  - E.g., a relay within `fbi.gov` tells you all input comes from `fbi.gov`

Circuits



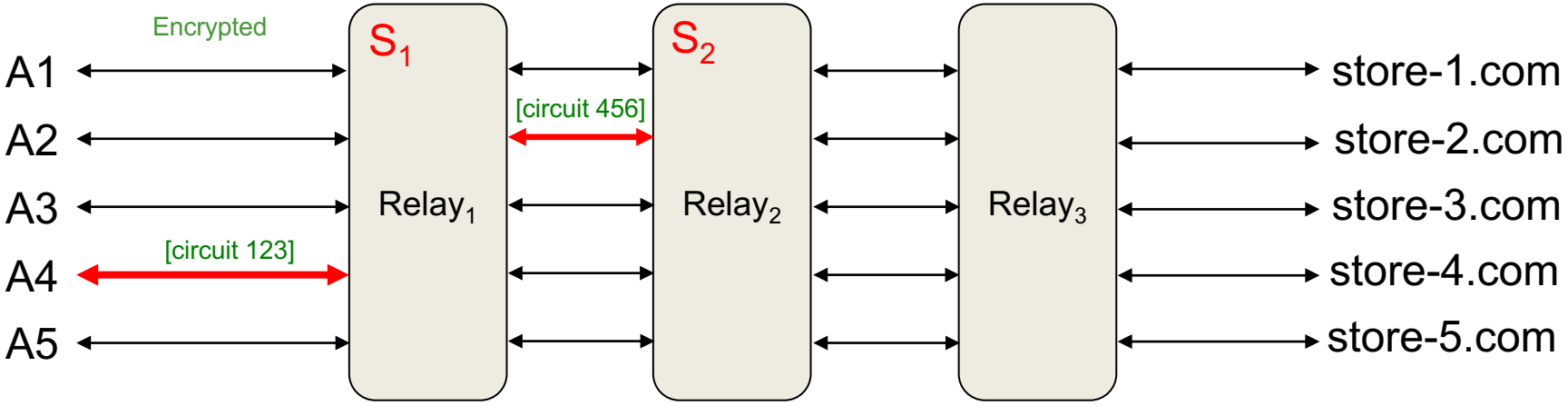
- Alice selects a list of relays through which her message will flow
- This path is called a circuit
- No node knows if the previous node is the originator or relay
  - Only the final node (**exit node**) knows it is the last node

Setting up a circuit (1)



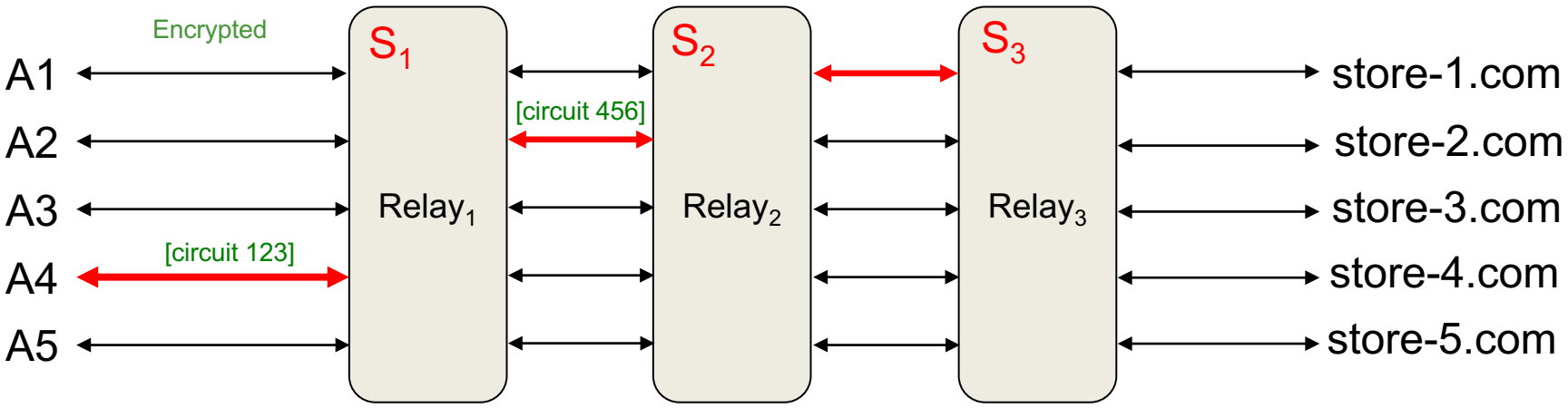
- Alice connects to Relay1
  - Sets up a TLS link to $Relay_1$
  - Does a one-way authenticated key exchange with **$Relay_1$** – agree on a symmetric key, **S_1**
  - Alice picks a circuit ID (e.g., 123) and asks Relay1 to create the circuit

Setting up a circuit (2)



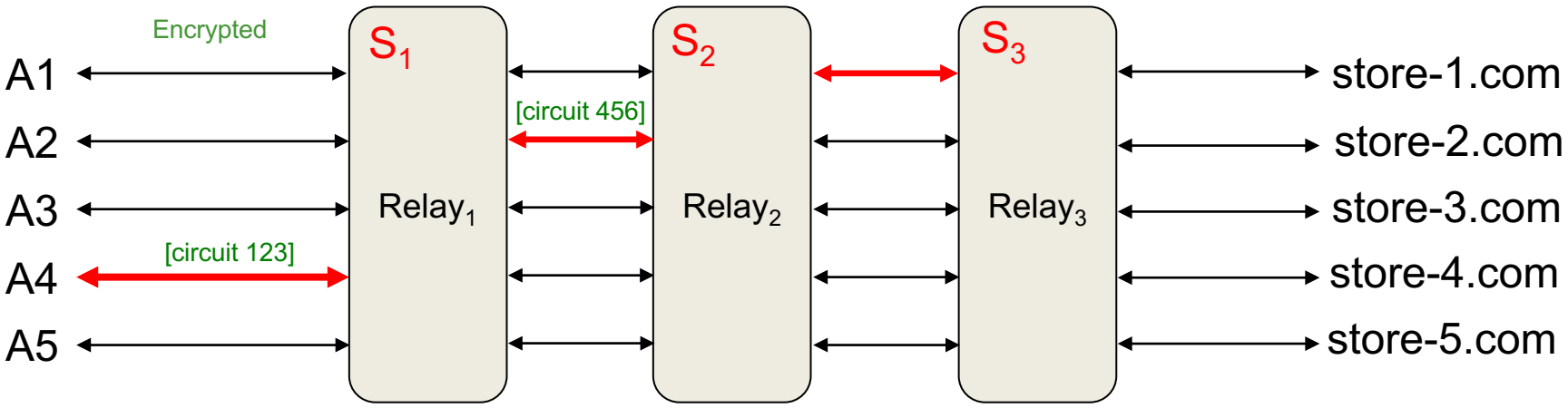
- Alice extends the relay to $Relay_2$
  - Alice sends a message to $Relay_1$:
    First part = "on circuit 1234, send ***Relay Extend*** to $Relay_2$ – the message is encrypted with S_1
  - $Relay_1$ establishes a TLS link to $Relay_2$ (if it didn't have one)
  - **Second part** of the message from Alice: initial handshake with $Relay_2$,
    encrypted with $Relay_2$'s public key
  - $Relay_2$ picks a random circuit for identifying this data stream to $Relay_2$, e.g., 456
    - Circuit 123 on $Relay_1$ connects to Circuit 456 on $Relay_2$
  - Does a one-way authenticated key exchange with **$Relay_2$** – agree on a symmetric key, **S_2**
    - All traffic flows through **$Relay_1$** and is encrypted with **S_1**

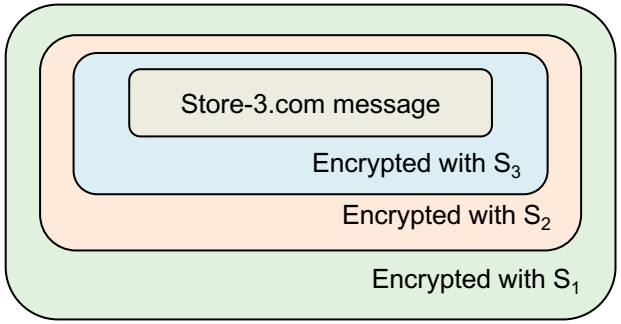
Setting up a circuit (3)



- Alice extends the relay to $Relay_3$
  - Same process – Alice sends a ***Relay Extend*** message to $Relay_2$
  - Messages to $Relay_2$ are encrypted with S_2 and then with S_1
    $E_{S1}(E_{S2}(M))$
  - $Relay_1$ decrypts the message to identify its circuit (123)
  - Routes message to $Relay_2$ on circuit 456
    - Circuit 123 is connected to circuit 456

Sending a message (5)





- Alice sends a message to store-3.com
- Each router strips off a layer of encryption
- At the end:
  - Directive to S_3 to open a TCP connection to store-3.com
  - Send messages
  - Get responses

Not a VPN

- Neither IP nor TCP packets are transmitted – just data streams
  - Too easy to identify the type of system by looking at TCP formats and responses
- Just take contents of TCP streams and relay the data
- End-to-end TLS works fine
  - TLS sits on top of TCP … it's just data going back and forth

Finding nodes

- Ideally, everyone would use some of the same nodes
  - Otherwise traffic would be distinguishable
- Multiple trusted parties supply node lists
  - Merge lists together
    - Union: if popularity-based, danger of someone flooding a list of nodes to capture traffic
    - Intersection: someone can block out nodes
  - Multiple parties vote on which nodes are running and behaving well
    - Distributed consensus
- Clients get
  - List of nodes and their public keys

Is it anonymous?

- Not really
- You can do a correlation attack
  - ISPs know who's talking to whom
  - May need to access traces from multiple ISPs
  - Can be really difficult if nodes have a lot of traffic (and it's similarly dense)
- Compromised exit node
  - Exit node decrypts the final layer and contacts the service

I2P and Garlic Routing

I2P = Invisible Internet Project

- Tor uses "onion routing"
  - Each message from the source is encrypted with one layer for each relay
- Garlic routing
  - Combines multiple messages at a relay
  - All messages, each with its own delivery instructions going to one relay are bundled together
  - Makes traffic analysis more difficult
- Tor circuits are bidirectional: responses take the same path
- I2P "tunnels" are unidirectional
  - One for outbound and one for inbound: the client builds both
  - Sender gets acknowledgement of successful message delivery

Services on top of I2P

- I2PTunnel: TCP connectivity
- Chat via IRC (Internet Relay Chat)
- File sharing
  - BitTorrent
  - iMule (anonymous file sharing)
  - I2Phex: Gnutella over I2P
- I2P-Bote: decentralized, anonymized email
  - Messages signed by the sender's private key
  - Anonymity via I2P and variable-rate delays
  - Destinations are I2P-Bote addresses
- I2P-Messenger, I2P-Talk
- Syndie: Content publishing (blogs, forums)

- Tor: far more users → more anonymity
  - Focused on anonymous access to services
- I2P: focuses on anonymous hosting of services
  - Uses a distributed hash table (DHT) for locating information on servers and routing
  - Server addressing
    - Uses cryptographic ID to identify routers and end services

The end