

# Lecture 11

## October 21, 2024

# Meaning of Identity

- Authentication validates identity
  - CA specifies type of authentication
  - If incorrect, CA may misidentify entity unintentionally
- Certificate binds *external* identity to crypto key and Distinguished Name
  - Need confidentiality, integrity, anonymity
    - Recipient knows same entity sent all messages, but *not* who that entity is

# Persona Certificate

- Certificate with meaningless Distinguished Name
  - If DN is  
/C=US/O=Microsoft Corp./CN=Bill Gates/  
the real subject may not (or may) be Mr. Gates
  - Issued by CAs with persona policies under a PCA with policy that supports this
- PGP certificates can use any name, so provide this implicitly

# Example

- Government requires all citizens with gene X to register
  - Anecdotal evidence people with this gene become criminals with probability 0.5.
  - Law to be made quietly, as no scientific evidence supports this, and government wants no civil rights fuss
- Government employee wants to alert media
  - Government will deny plan, change approach
  - Government employee will be fired, prosecuted
- Must notify media anonymously

# Example

- Employee gets persona certificate, sends copy of plan to media
  - Media knows message unchanged during transit, but not who sent it
  - Government denies plan, changes it
- Employee sends copy of new plan signed using same certificate
  - Media can tell it's from original whistleblower
  - Media cannot track back whom that whistleblower is

# Trust

- Goal of certificate: bind correct identity to DN
- Question: what is degree of assurance?
- X.509v4, certificate hierarchy
  - Depends on policy of CA issuing certificate
  - Depends on how well CA follows that policy
  - Depends on how easy the required authentication can be spoofed
- Really, estimate based on the above factors

# Example: Passport Required

- DN has name on passport, number and issuer of passport
- What are points of trust?
  - Passport not forged and name on it not altered
  - Passport issued to person named in passport
  - Person presenting passport is person to whom it was issued
  - CA has checked passport and individual using passport

# PGP Certificates

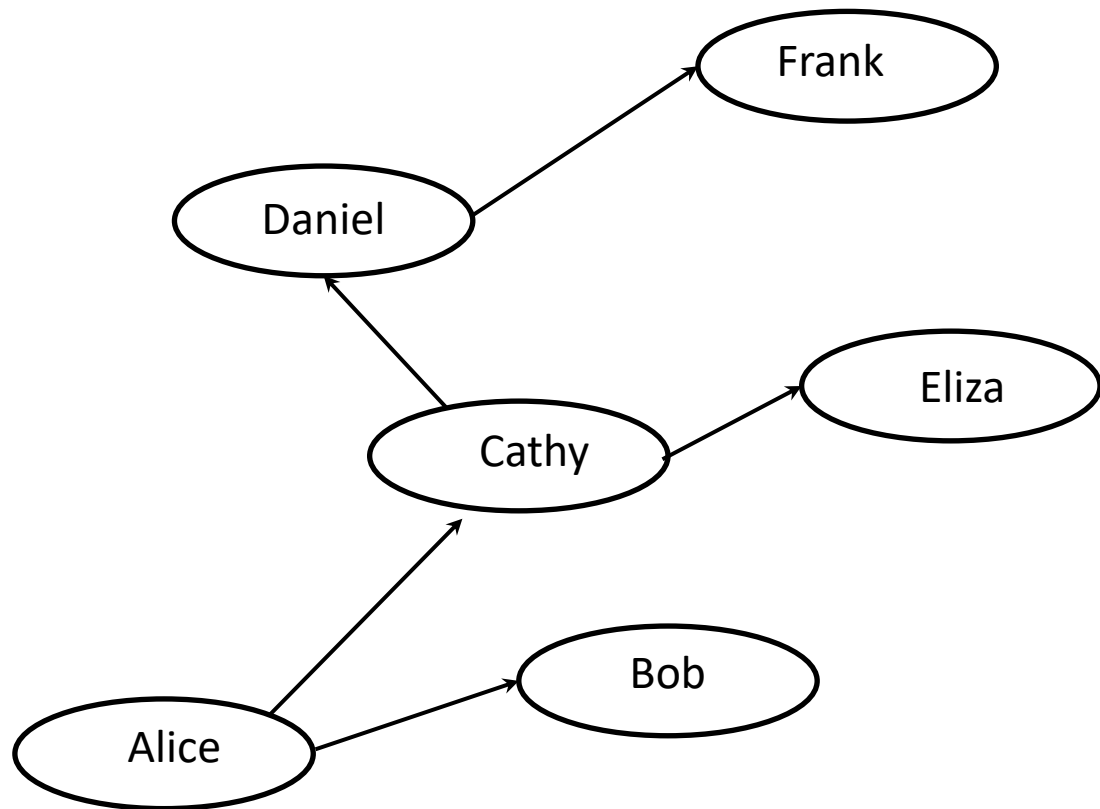
- Public key packet
  - Version
  - Time of creation
  - Validity period
  - Public key algorithm and parameters
  - Public key
- Followed by 0 or more signature packets
- Signature packet (OpenPGP v3)
  - Version
  - Signature type (trust level)
  - Creation time
  - Key identifier of the signer
  - Public key algorithm
  - Hash algorithm
  - Part of signed hash value
  - Signature



# PGP Certificates

- Level of trust in signature field signature type
- Four levels
  - Generic (no trust assertions made)
  - Persona (no verification)
  - Casual (some verification)
  - Positive (substantial verification)
- What do these mean?
  - Meaning not given by OpenPGP standard
  - Signer determines what level to use
  - Casual to one signer may be positive to another

# Web of Trust



Alice needs Frank's certificate

- She doesn't have it so she asks Bob and Cathy if they do
- Neither do, so Cathy asks Daniel and Eliza
- Daniel knows Frank and gets his public key
- Daniel decides how much he trusts Frank and that the certificate is Frank's, and forwards both to Cathy
- Daniel decides how much he trusts Frank and that the certificate is Frank's, and forwards both to Cathy
- Cathy decides how much she trusts Daniel, and forwards that and the certificate to Alice
- Alice decides whether to accept the certificate as legitimate or reject it.

Note: no certification or registration authorities needed

# Anonymity on the Web

- Recipients can determine origin of incoming packet
  - Sometimes not desirable
- Anonymizer: a site that hides origins of connections
  - Usually a proxy server
    - User connects to anonymizer, tells it destination
    - Anonymizer makes connection, sends traffic in both directions
  - Destination host sees only anonymizer

# Example: *anon.penet.fi*

Offered anonymous email service

- Sender sends letter to it, naming another destination
- Anonymizer strips headers, forwards message
  - Assigns an ID (say, 1234) to sender, records real sender and ID in database
  - Letter delivered as if from anon1234@anon.penet.fi
- Recipient replies to that address
  - Anonymizer strips headers, forwards message as indicated by database entry

# Problem

- Anonymizer knows who sender, recipient *really* are
- Called *pseudo-anonymous remailer* or *pseudonymous remailer*
  - Keeps mappings of anonymous identities and associated identities
- If you can get the mappings, you can figure out who sent what

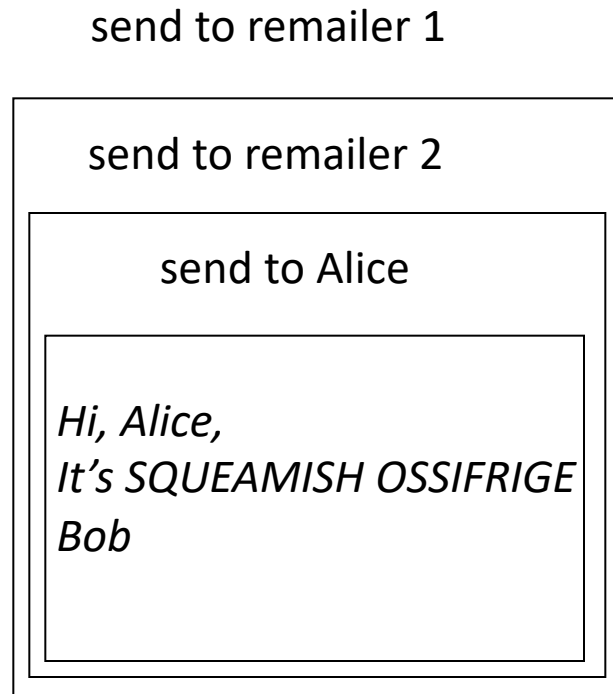
# More *anon.penet.fi*

- Material claimed to be copyrighted sent through site
- Finnish court directed owner to reveal mapping so plaintiffs could determine sender
- Owner appealed, subsequently shut down site

# Cypherpunk Remailer

- Remailer that deletes header of incoming message, forwards body to destination
- Also called *Type I Remailer*
- No record kept of association between sender address, remailer's user name
  - Prevents tracing, as happened with *anon.penet.fi*
- Usually used in a chain, to obfuscate trail
  - For privacy, body of message may be enciphered

# Cypherpunk Remailer Message



- Encipher message
- Add destination header
- Add header for remailer  $n$
- ...
- Add header for remailer 1



# Weaknesses

- Attacker monitoring entire network
  - Observes in, out flows of remailers
  - Goal is to associate incoming, outgoing messages
- If messages are cleartext, trivial
  - So assume all messages enciphered
- So use traffic analysis!
  - Used to determine information based simply on movement of messages (traffic) around the network

# Attacks

- If remailer forwards message before next message arrives, attacker can match them up
  - Hold messages for some period of time, greater than the message interarrival time
  - Randomize order of sending messages, waiting until at least  $n$  messages are ready to be forwarded
    - Note: attacker can force this by sending  $n-1$  messages into queue

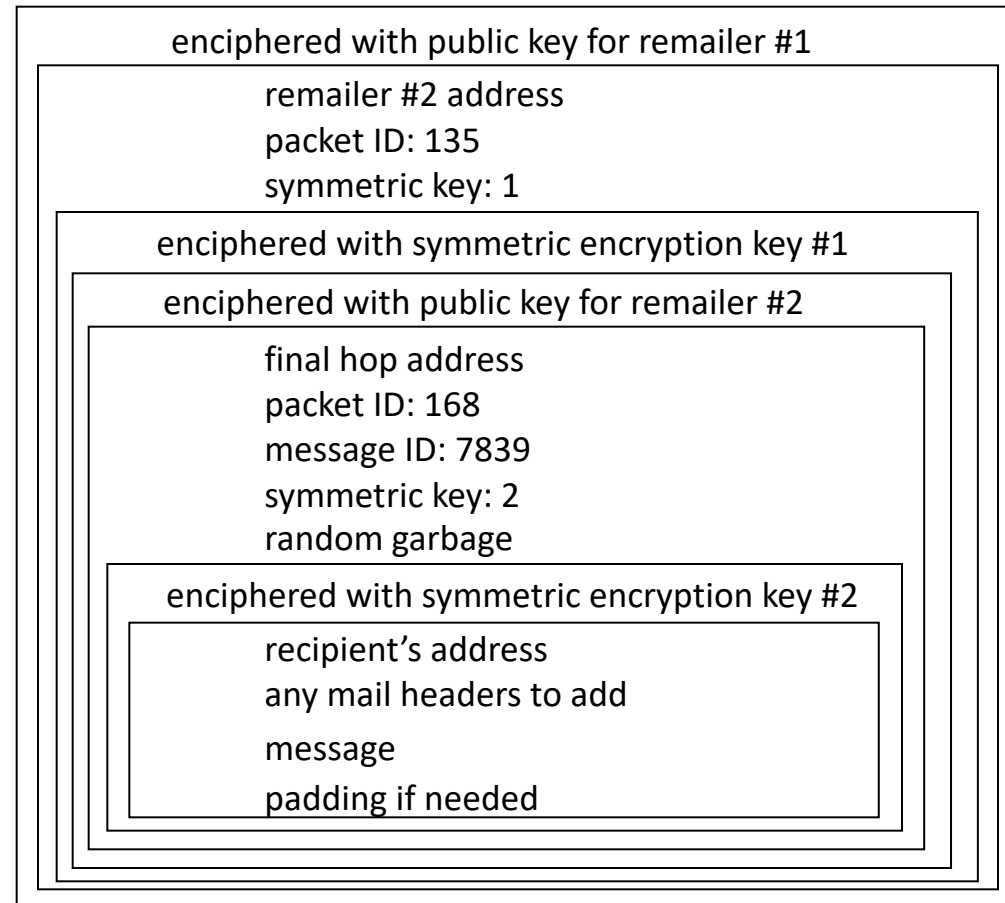
# Attacks

- As messages forwarded, headers stripped so message size decreases
  - Pad message with garbage at each step, instructing next remailer to discard it
- Replay message, watch for spikes in outgoing traffic
  - Remailer can't forward same message more than once

# Mixmaster (Cypherpunk Type 2) Remailer

- Cypherpunk remailer that handles only enciphered mail and pads (or fragments) messages to fixed size before sending them
  - Also called Type 2 Remailer
  - Designed to hinder attacks on Cypherpunk remailers
    - Messages uniquely numbered
    - Fragments reassembled *only* at last remailer for sending to recipient

# Cypherpunk Remailer Message



# Onion Routine

- Method of routing so each node in the route knows only the previous and following node
  - Typically, first node selects the route
  - Intermediate node may be able to change rest of route
- Each intermediate node has public, private key pair
  - Public key available to all nodes and any proxies
- Client, server have proxies to handle onion routing

# Heart of the Onion Route

$\{ \textit{expires} \parallel \textit{nexthop} \parallel E_F \parallel k_F \parallel E_B \parallel k_B \parallel \textit{payload} \} \textit{pub}_r$

- *payload*: data associated with message
- *expires*: expiration time for which *payload* is to be saved
- *nexthop*: node to forward message to
- $\textit{pub}_r$ : public key of next hop (node)
- $E_F, k_F$ : encryption algorithm, key to be used when sending message forward to server
- $E_B, k_B$ : encryption algorithm, key to be used when sending message backwards to client

# Notes About the Heart

- *payload* may itself be a message of this form or the data being sent
- Each router has table storing:
  - Virtual circuit number associated with a route
  - $E_F, k_F, E_B, k_B$  for the next, previous nodes on the route
  - Next router to which messages using this route are to be forwarded
    - If last router on route, this is NULL (as is *nexthop* in the packet)



# Creating a Route

- Client's proxy determines route for the message
  - Can be defined exactly, or loosely, where the intermediate routers can route messages to next hop over other routes
- Create onion encapsulating route, put it in a *create* message and add virtual circuit number
- Forward to next (second) router on path
- That router deciphers the onion using its private key ("peeling the onion")
  - Compare it to what's in table; if replay, discard

# Creating a Route

- Router creates new virtual circuit number, and add to table:
  - (virtual circuit number in message, created virtual circuit number) pair
  - Keys, algorithms in onion
- Router generates new *create* message, puts assigned virtual circuit number and “peeled” onion in it
  - This is smaller than the onion received, so add padding to make it the same size
- Forward it to next hop

# Sending a Message

- Sender applies decryption algorithms corresponding to each backwards encryption algorithm along the route
- Example: route begins at  $W$ , then through  $X$  and  $Y$  to  $Z$ ;  $W$  constructs this:

$$d_x(k_x, d_y(k_y, d_z(k_z, m)))$$

- Sends this to  $X$ , which uses its  $E_B$  to encrypt message, getting  $d_y(k_y, d_z(k_z, m))$
- Forwards this to  $Y$ , which uses its  $E_B$  to encrypt message, getting  $d_z(k_z, m)$
- Forwards this to  $Z$ , which uses its  $E_B$  to encrypt message, getting  $m$

# Potential Attacks

- If client's proxy compromised, attacker can see all routes selected and all messages, and so may be able to deduce server
- If server's proxy compromised, attacker can see all messages but cannot deduce the routes
- If router compromised, attacker can determine only the previous, next routers in path
  - In particular, the attacker cannot read the encrypted onion
- If attacker can see all traffic on network, they can
  - Match client, server message sizes; that's why all messages are padded to same size
  - Observe the flow of messages; that's why the onion network sends meaningless messages to obscure that flow

# Example: Tor (The Onion Router)

- Connects clients, servers over virtual circuits set up among onion routers (*OR*)
  - Each OR has identity key, onion key
  - Identity key signs information about router
  - Onion key used to read requests to set up circuits; changed periodically
  - All virtual circuits over TLS, and a third TLS key established for this
- Basic message unit: *cell*, always 512 bytes long
  - Control cell: header contains command directing recipient to do something
    - Create a circuit, circuit created, destroy a circuit
  - Relay cell: deals with an established circuit
    - Open stream, stream opened, extend circuit, circuit extended, close stream cleanly, close broken stream, cell contains data

# Setting Up Virtual Circuit

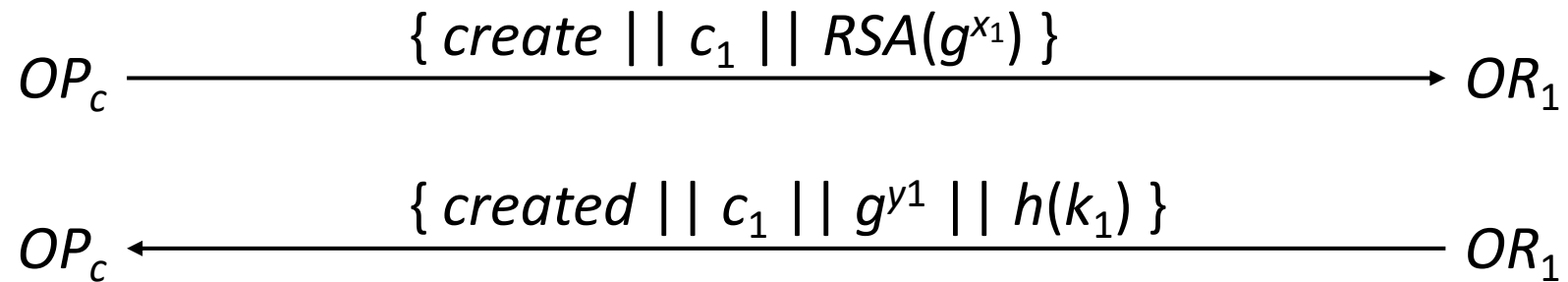
- Set up over TLS connections
  - Several circuits may use same TLS connection to reduce overhead
- Streams move data over virtual circuits
  - Several streams may be multiplexed over one circuit
- Client's onion proxy  $OP_c$  needs to know where ORs are
  - Tor uses directory services for this; group of well-known ORs track information about usable ORs, including keys, addresses
  - $OP_c$  contacts one such directory server, gets information from it, chooses path

# Setting Up Virtual Circuit

- Tor uses 3 ORs ( $OR_1, OR_2, OR_3$ ); client, server proxies  $OP_c, OP_s$
- $RSA(x)$  is enciphering of message  $x$  using onion key of destination OR
- $g, p$  as in Diffie-Hellman
- $x_1, \dots, x_n$  and  $y_1, \dots, y_n$  generated randomly;  $k_i = g^{x_i y_i} \text{ mod } p$ , and forward, backwards keys selected from this
- $h(x)$  cryptographic hash of  $x$
- All links are over TLS and so encrypted (TLS keys not shown on next slide)

# Tor Protocol to Create Virtual Circuit

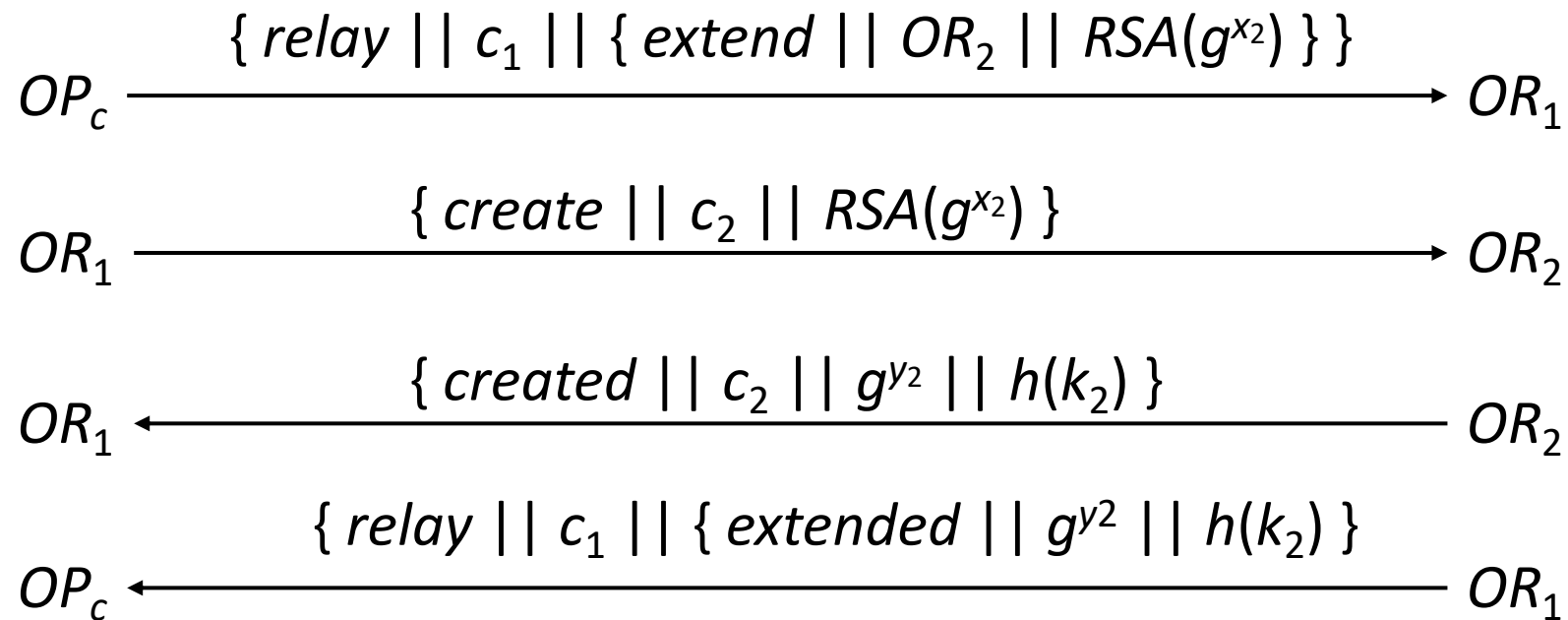
This sets up the part of the virtual circuit between  $OP_c$  and  $OR_1$ :





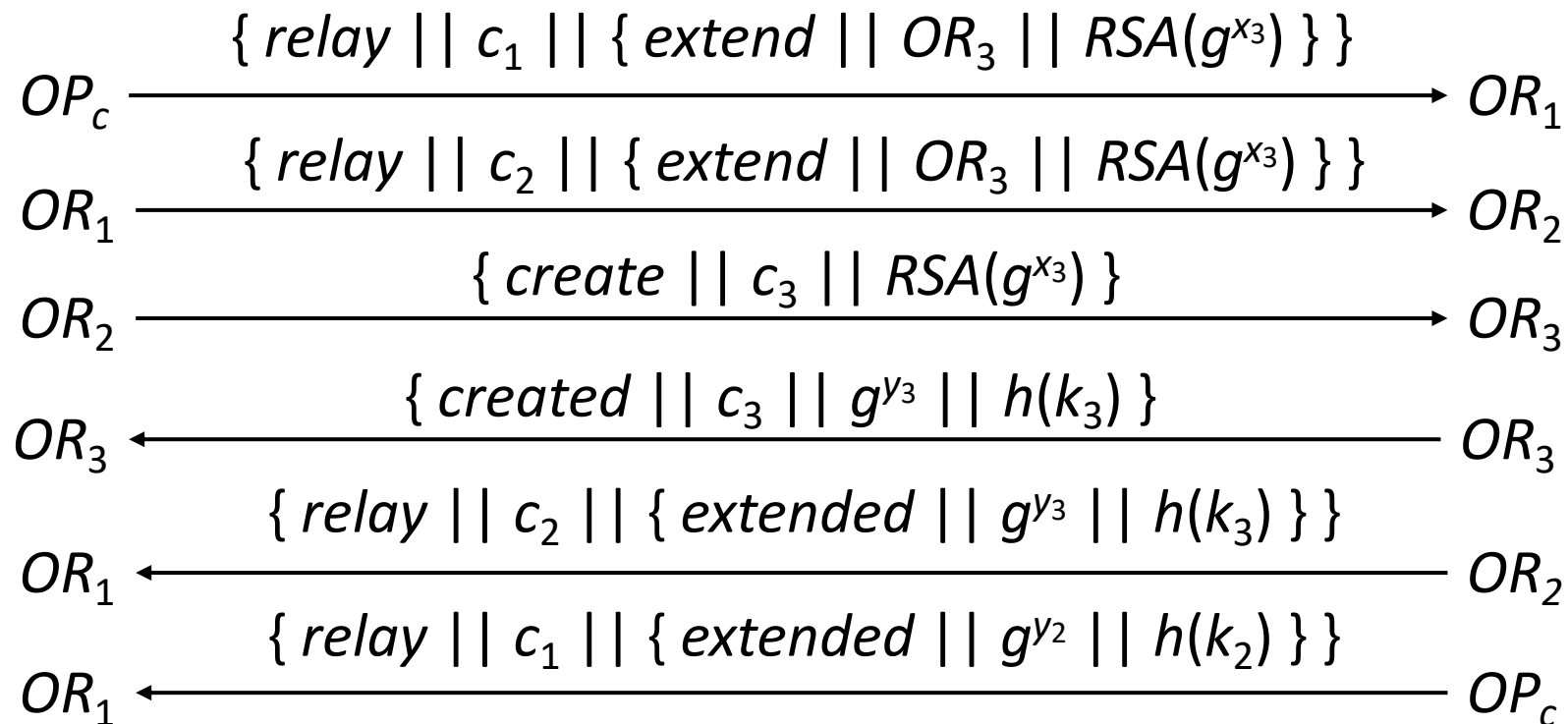
# Tor Protocol to Create Virtual Circuit

This sets up the part of the virtual circuit between  $OP_c$  and  $OR_2$ :



# Tor Protocol to Create Virtual Circuit

This sets up the part of the virtual circuit between  $OP_c$  and  $OR_3$ :



# After All This . . .

- $OP_c$  has forward keys for  $OR_1, OR_2, OR_3$ ; call them  $f_1, f_2, f_3$ 
  - Here,  $f_i = g^{y_i} \text{ mod } p$
- To send message  $m$  to server, client sends  $m$  to  $OP_c$ 
  - $OP_c$  enciphers it using AES-128 in counter mode, getting  $\{ \{ \{ m \}_{f_1} \}_{f_2} \}_{f_3}$
  - It puts this into a relay cell and sends it to  $OR_1$
- $OR_1$  deciphers cell, determines next hop by looking up virtual circuit number in its table, puts  $\{ \{ m \}_{f_1} \}_{f_2}$  into another relay cell, forwards it to  $OR_2$
- $OR_2$  does same, but forwards it to  $OR_3$
- $OR_3$  deciphers cell, either does what  $m$  requests (eg, open TLS connection to server) or forwards payload  $m$  to server

# Server Replies

- Server sends reply  $r$  to  $OR_3$
- $OR_3$  enciphers it using its backwards key, embeds it in relay cell, forwards it to  $OR_2$
- $OR_2$  uses circuit number to determine  $OR_1$ , enciphers cell using its backwards key, forwards it to  $OR_1$
- $OR_1$  does same but forwards it to  $OP_c$
- $OP_c$  has all the forward keys, and so can decipher the message and forward it to client

# Use Problems

Adversary wants to determine who is using onion routing network

- Attack: monitor the client, known entry router
  - Solution: use unlisted entry routers
  - Example: Tor uses *bridge relays* that are not listed in Tor directories; to find them, go to specific web page or email a specific set of addresses; result is a list of entry routers (bridges) that  $OP_c$  can use
- Attack: examine packets sent from a client looking for structures indicating that they are intended for onion routers
  - Solution: obfuscate packet contents; endpoint deobfuscates it
  - Example: Tor has *pluggable transports* that do this

# Anonymity Itself

- Some purposes for anonymity
  - Removes personalities from debate, or with appropriate choice of pseudonym, shape course of debate by implication
  - Prevent retaliation
  - Protect privacy
- Are these benefits or drawbacks?
  - Depends on society, and who is involved

# Pseudonyms

- Names of authors of documents used to imply something about the document
- Example: *U.S. Federalist Papers*
  - These argued for the states adopting the U.S. Constitution
  - Real authors were Alexander Hamilton, James Madison, John Jay, all Federalists who wanted the Constitution adopted
  - But using alias “Publius” hid their names
    - Debate could focus on content of the *Federalist Papers*, not the authors or their personalities
    - Roman Publius seen as a model governor, implying the *Papers* represented responsible political philosophy, legislation

# Whistleblowers

- Criticism of powerholders often fall into disfavor; powerholders retaliate, but anonymity protects these critics
  - Example: Anonymous sources spoke to Woodward and Bernstein, during U.S. Watergate scandal in 1970s; one important source, called “Deep Throat”, provided guidance that helped uncover a pattern of activity leading to impeachment articles against President Nixon and his resignation
    - “Deep Throat” later revealed as an assistant director of Federal Bureau of Investigation; had this been known, he would have been fired and might have been prosecuted
  - Example: Galileo openly held Copernican theory of the earth circling the sun; brought before the Inquisition and forced to recant



# Privacy

- Anonymity protects privacy by obstructing amalgamation of individual records
- Important, because amalgamation poses 3 risks:
  - Incorrect conclusions from misinterpreted data
  - Harm from erroneous information
  - Not being let alone
- Also hinders monitoring to deter or prevent crime
- Conclusion: anonymity can be used for good or ill
  - Right to remain anonymous entails responsibility to use that right wisely