Security and privacy in networks

Damien Saucez
Inria Sophia Antipolis

Université d’Avignon et des Pays de Vaucluse, CERI - Securité dans les réseaux - March 2018

Contact information

- Damien Saucez
  - Email: damien.saucez@inria.fr

Table of Content

1. Reminders
2. Threats by the example
3. The basics of security
4. Securing communications
5. Blockchains
6. Privacy
7. Overlay networking

References

- O. Bonaventure. Computer Networking: Principles, Protocols and Practice. http://inl.info.ucl.ac.be/CNP3.
    - some network slides inspired from this book
- J. Kurose and K. Ross. Computer Networking: A Top-Down Approach, Addison-Wesley, 6th Edition.
- L. Peterson and B. Davie.
  Computer Networks: A Systems Approach.
  Morgan Kaufmann Publishers, 4th Edition.
- A. Tanenbaum, D. Wetherall, Computer Networks, Prentice Hall, 4th Edition
- A. Legout, Peer-to-Peer Applications From BitTorrent to Privacy, Inria
    - some privacy and peer-to-peer slides inspired from this course
- J. Kehrli, The Blockchain – The Technology behind Bitcoin, https://www.slideshare.net/JrmeKehrli/the-blockchain-the-technology-behind-bitcoin, November 2017
    - some blockchain slides inspired from this presentation

Reminders

Naming and addressing

Addressing in Ethernet

- Objective: determine the origin and destination of a frame within a collision domain
- Every Ethernet network adapter is assigned a unique datalink layer address encoded on 48 bits
- Every frame is transmitted to all network adapters of the collision domain
    - but only the network adapter with the address corresponding to the destination address of the frame accepts it

Addressing in IP

- Objective: determine the origin and destination of a packet in the Internet
- Every host interface has its own IP address
  - routers have multiple interfaces, each with its own IP address
  - the IP address determines the topological position of the interface
- Current version of IP is version 4 (IPv4)
  - addresses are encoded on 32 bits, fixed length
- 4 billions addresses were a lot... in 1981, but is way too short today
- IP version 6 (IPv6) starts to be deployed
  - addresses are encoded on 128 bits, fixed length*

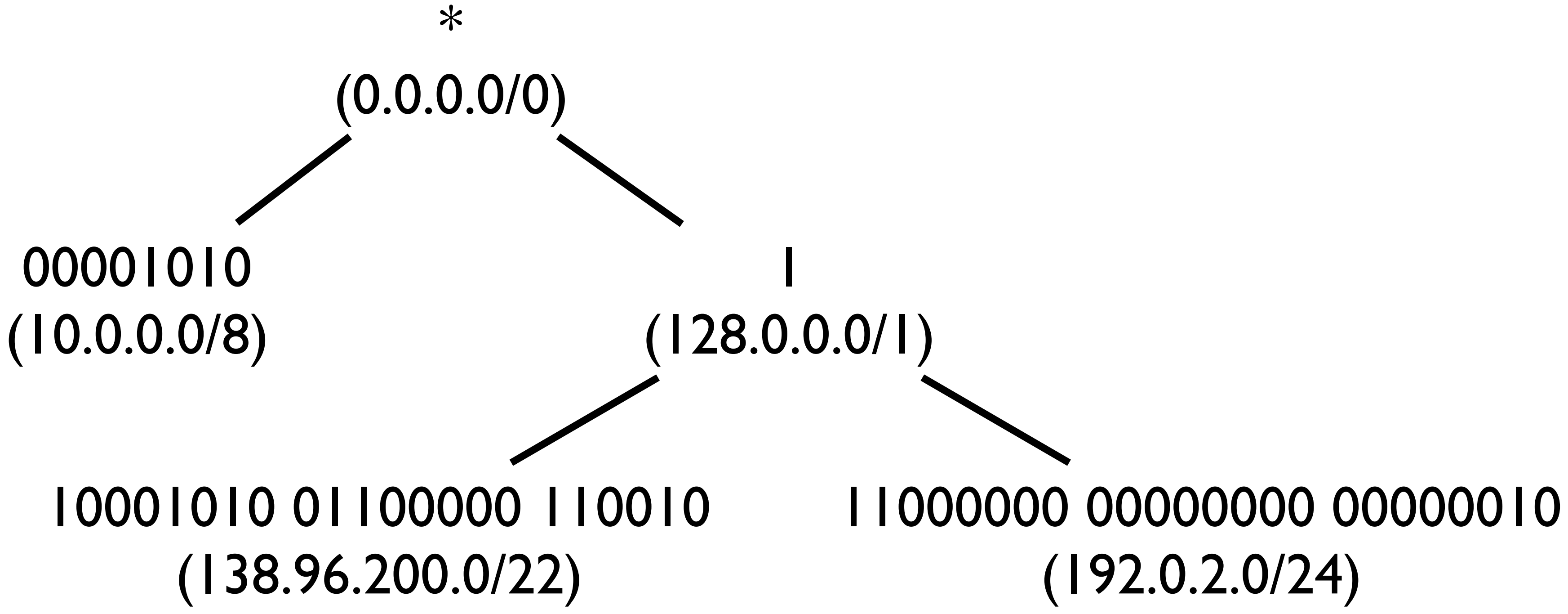
Classless InterDomain Routing (CIDR)

- No predetermined separation position between network number and local address with CIDR
  - number of bits allocated for the network number may vary from 0 to 32 (resp. 128) bits in IPv4 (resp. IPv6)
  - the address contains no information about the separation position
  - Routers determine the network number by using longest-prefix matching
- Notation *a.b.c.d*/*n* (resp. *a:b:c:d:e:f:g:h/n*)
  - *a.b.c.d* (resp. *a:b:c:d:e:f:g:h*) is the address
  - *n* is the number of bits assigned to the network number

CIDR (cont.)

- An address matches a route if both share the same prefix
  - 0.0.0.0/0 (resp. ::/0) is the default route matched by every addresses
- With CIDR, an address can match several routes
  - 192.0.2.1 matches 128.0.0.0/1, but also 192.0.2.0/24 or 0.0.0.0/0
- Longest prefix matching is used to determine the route that has the longest prefix in common with the address
- Typically implemented with a trie

Longest prefix matching with a trie (examples)



*
(0.0.0.0/0)

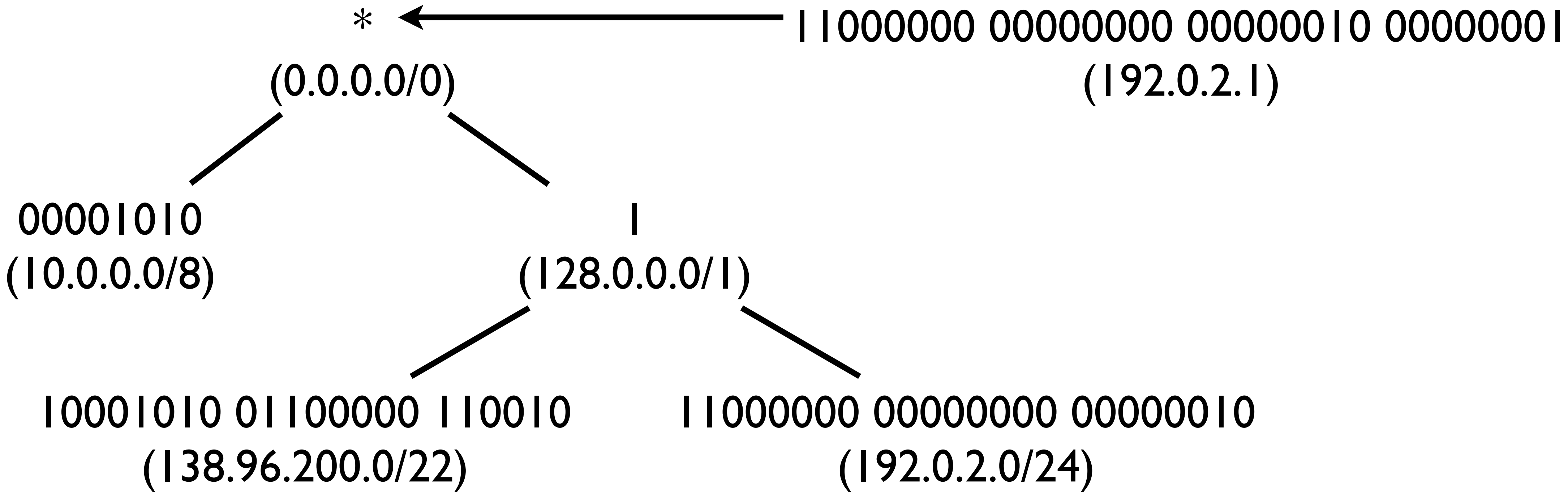
00001010
(10.0.0.0/8)

1
(128.0.0.0/1)

10001010 01100000 110010
(138.96.200.0/22)

11000000 00000000 00000010
(192.0.2.0/24)

Longest prefix matching with a trie (examples)



*
(0.0.0.0/0)

11000000 00000000 00000010 00000001
(192.0.2.1)

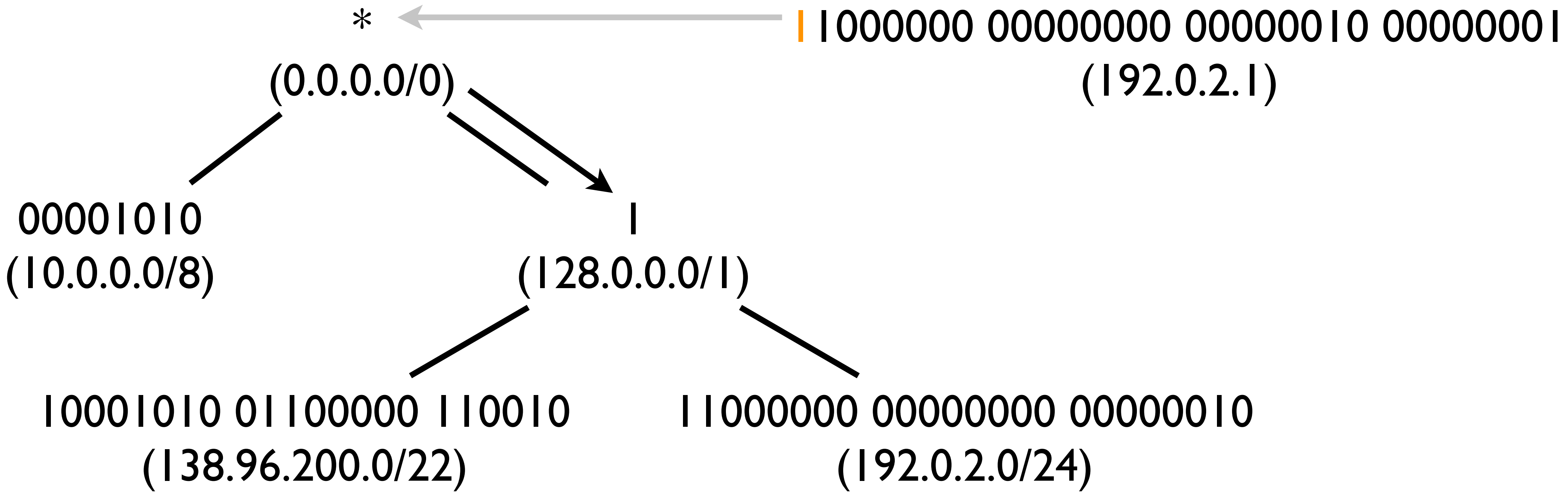
00001010
(10.0.0.0/8)

1
(128.0.0.0/1)

10001010 01100000 110010
(138.96.200.0/22)

11000000 00000000 00000010
(192.0.2.0/24)

Longest prefix matching with a trie (examples)



*
(0.0.0.0/0)

11000000 00000000 00000010 00000001
(192.0.2.1)

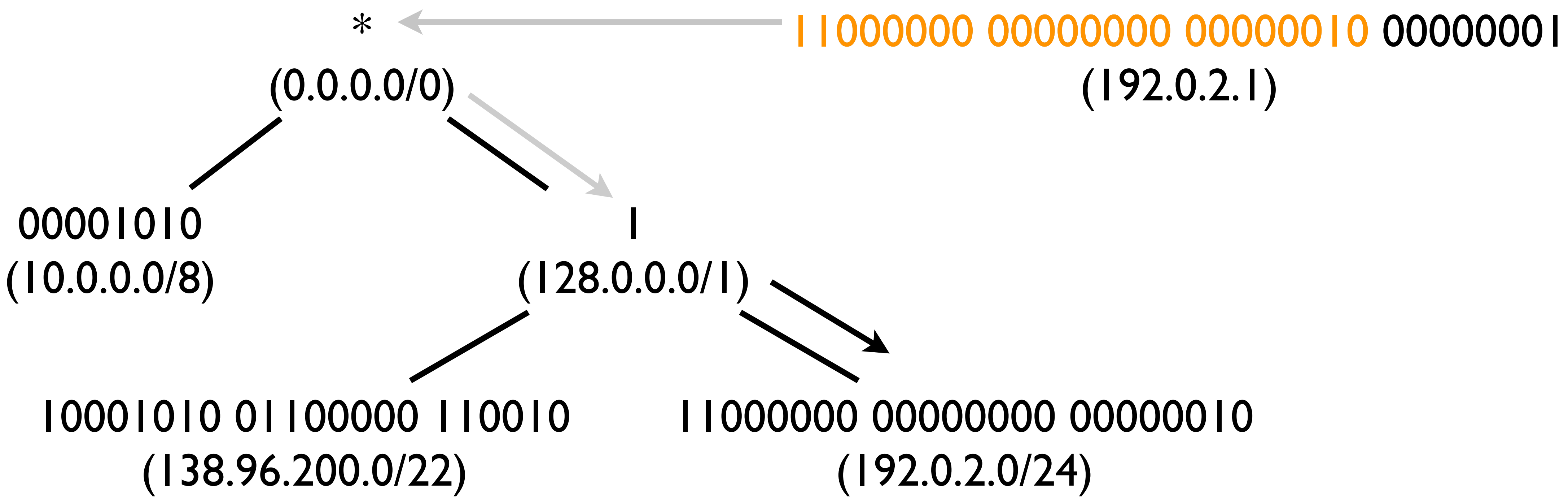
00001010
(10.0.0.0/8)

1
(128.0.0.0/1)

10001010 01100000 110010
(138.96.200.0/22)

11000000 00000000 00000010
(192.0.2.0/24)

Longest prefix matching with a trie (examples)



*
(0.0.0.0/0)

11000000 00000000 00000010 00000001
(192.0.2.1)

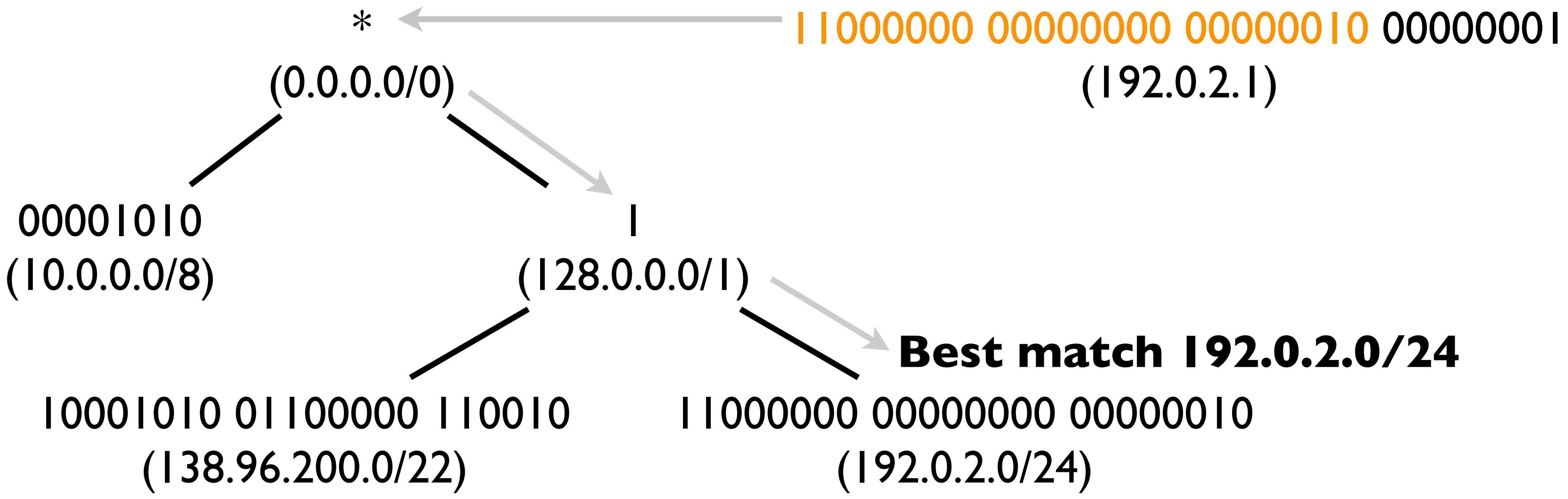
00001010
(10.0.0.0/8)

1
(128.0.0.0/1)

10001010 01100000 110010
(138.96.200.0/22)

11000000 00000000 00000010
(192.0.2.0/24)

Longest prefix matching with a trie (examples)



*

(0.0.0.0/0)

11000000 00000000 00000010 00000001

(192.0.2.1)

00001010

(10.0.0.0/8)

1

(128.0.0.0/1)

Best match 192.0.2.0/24

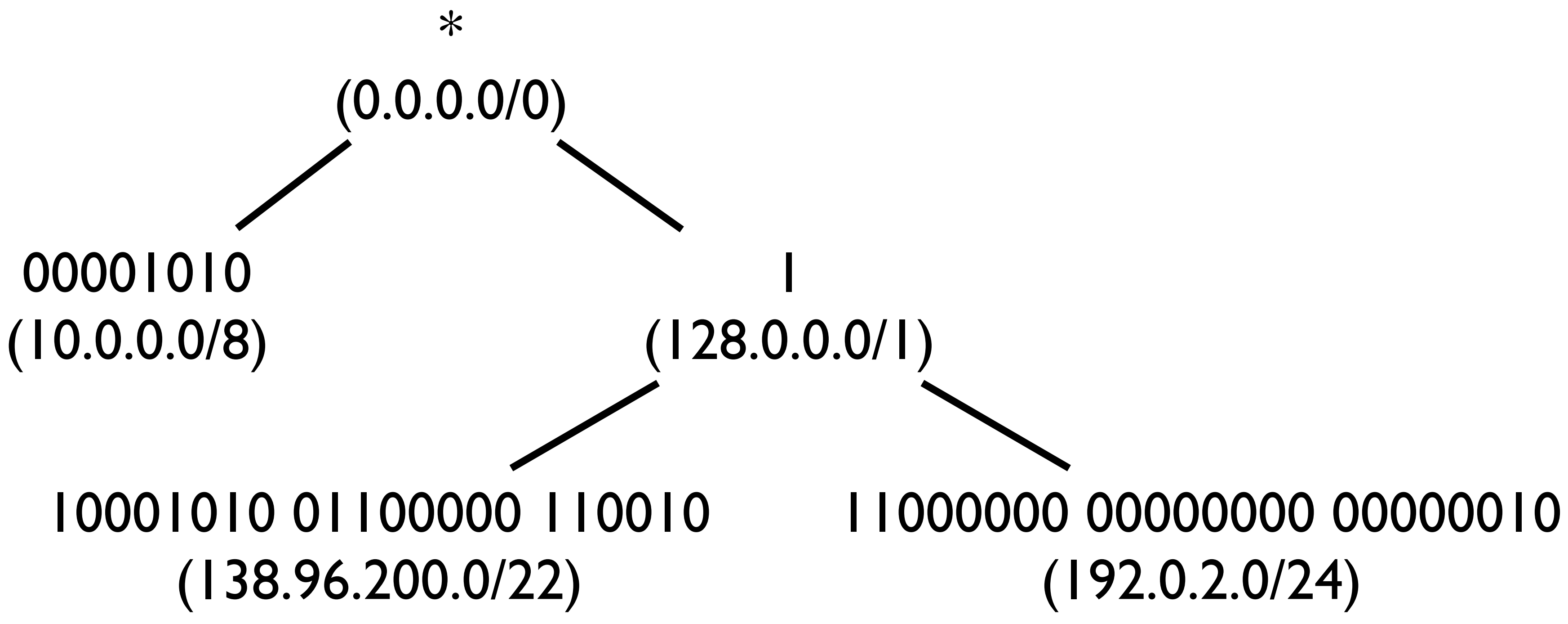
10001010 01100000 110010

(138.96.200.0/22)

11000000 00000000 00000010

(192.0.2.0/24)

Longest prefix matching with a trie (examples)



*

(0.0.0.0/0)

00001010

(10.0.0.0/8)

1

(128.0.0.0/1)

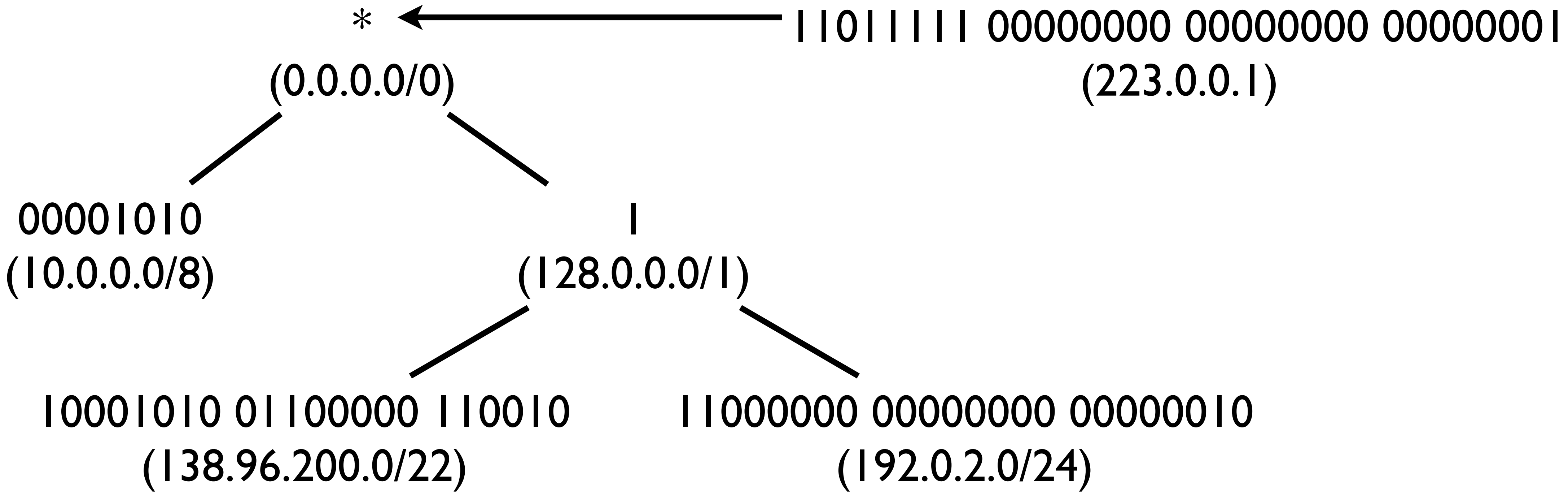
10001010 01100000 110010

(138.96.200.0/22)

11000000 00000000 00000010

(192.0.2.0/24)

Longest prefix matching with a trie (examples)



*

(0.0.0.0/0)

11011111 00000000 00000000 00000001

(223.0.0.1)

00001010

(10.0.0.0/8)

1

(128.0.0.0/1)

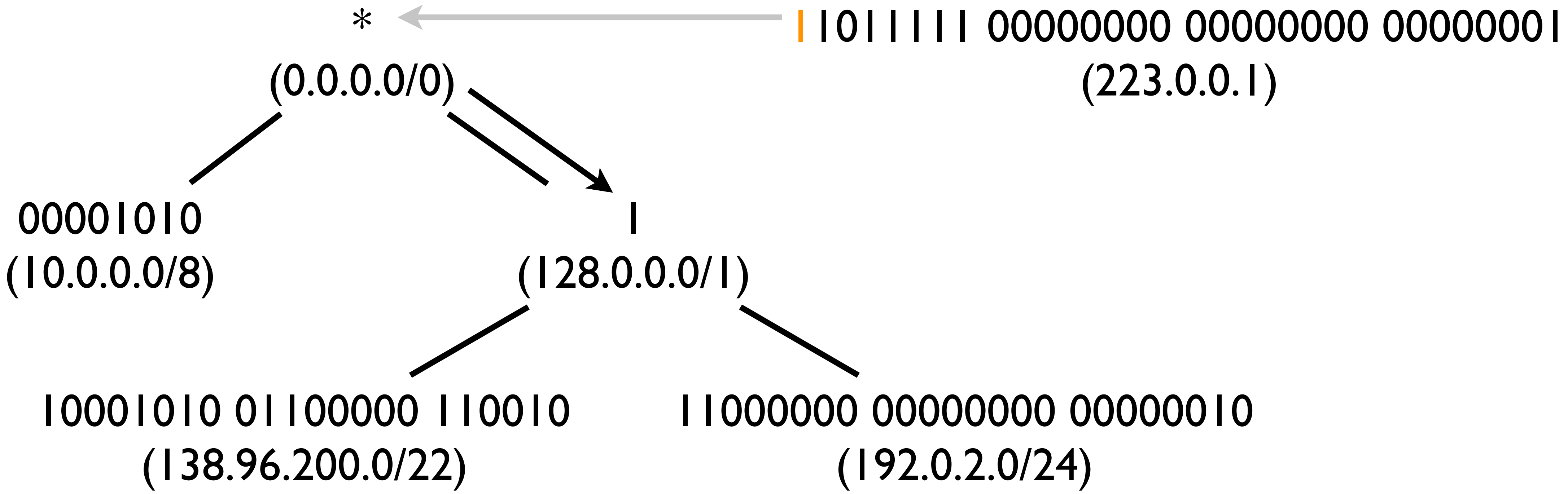
10001010 01100000 110010

(138.96.200.0/22)

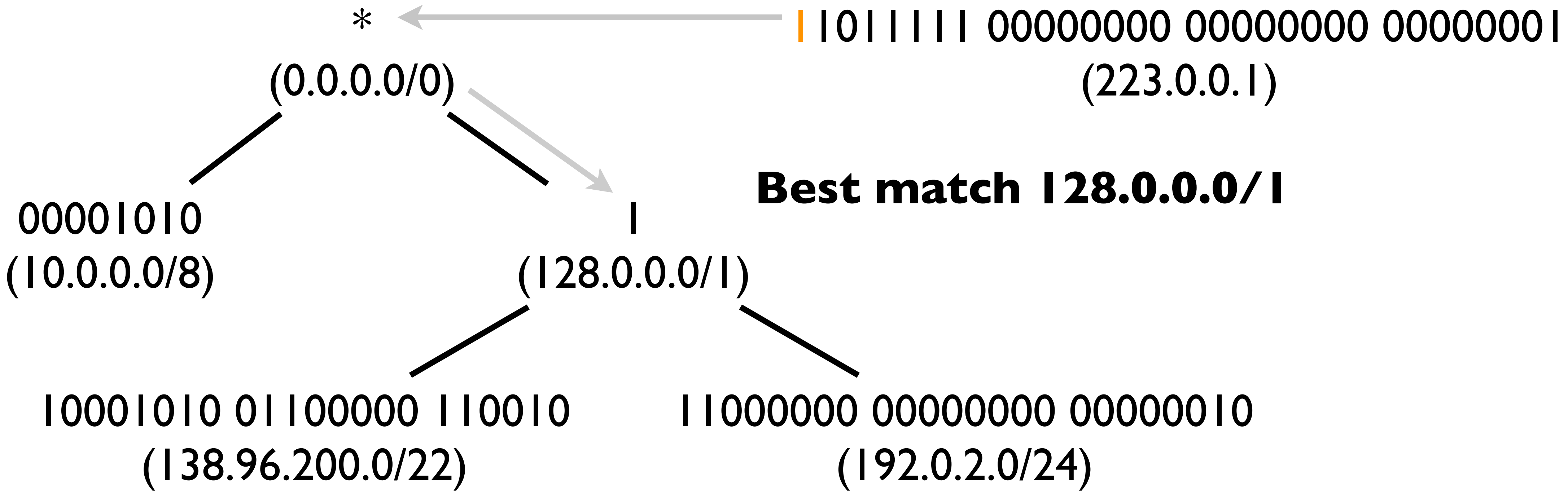
11000000 00000000 00000010

(192.0.2.0/24)

Longest prefix matching with a trie (examples)



Longest prefix matching with a trie (examples)



*
(0.0.0.0/0)

1 1011111 00000000 00000000 00000001
(223.0.0.1)

00001010
(10.0.0.0/8)

1
(128.0.0.0/1)

10001010 01100000 110010
(138.96.200.0/22)

11000000 00000000 00000010
(192.0.2.0/24)

Best match 128.0.0.0/1

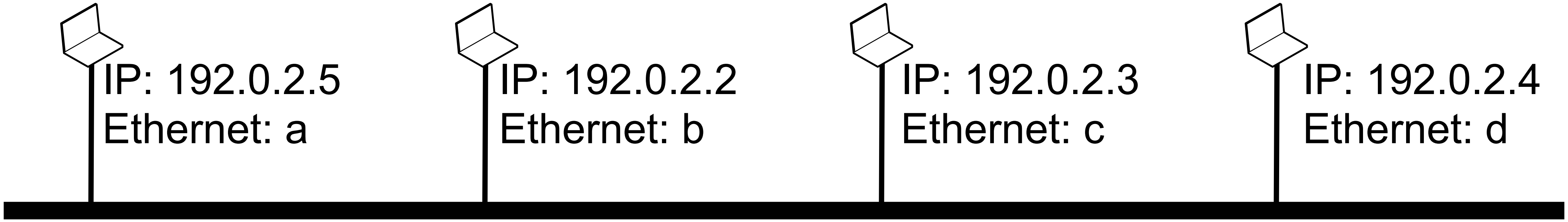
IP to Ethernet Address

- To put an IP packet over an Ethernet frame, its IP addresses must be resolved into Ethernet addresses
- Protocol used:
  - Address Resolution Protocol (ARP) in IPv4
  - Neighbor Discovery Protocol (NDP) in IPv6

ARP

- ARP is used to get datalink layer address of a machine on the local subnet
- Broadcast an ARP request frame on the local subnet for the IP address to resolve
  - destination address: FF:FF:FF:FF:FF:FF (broadcast)
  - source address: Ethernet address of the network adapter that issued the ARP request
- The host (or a proxy) that owns the address replies with an ARP response frame
  - destination address: Ethernet address of the requester's network adapter
  - source address: Ethernet address of the address's owner's (or proxy) network adapter
- Every network device is required to listen for ARP requests and replies on its network adapters
- Optimizations
  - replies are stored in an ARP cache to avoid that every single packet results in ARP request/ response
    - cached for a limited duration as host can change their IP address
  - ARP request message contains the IP address of the origin of the frame
    - destination (or any hosts in the local subnet) can learn the IP/Ethernet mapping for free

ARP example



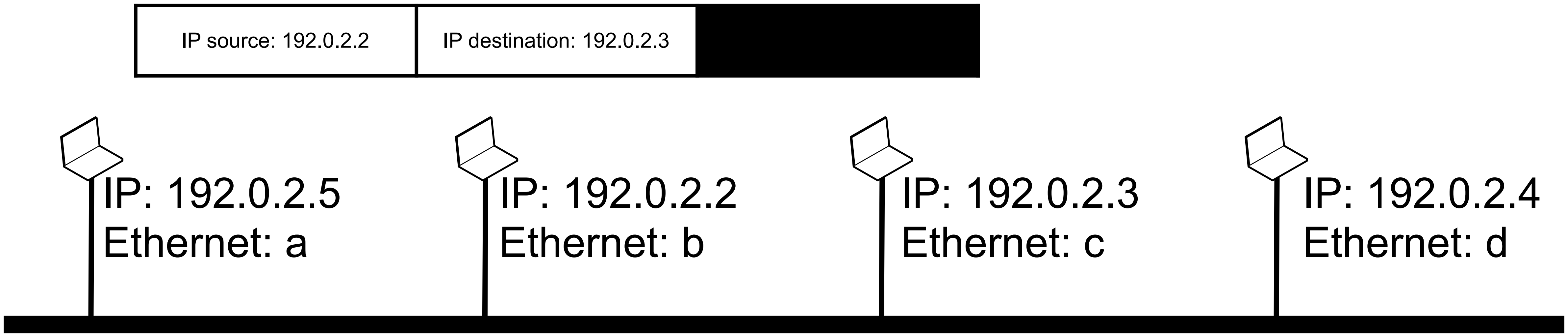
IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.3
Ethernet: c

IP: 192.0.2.4
Ethernet: d

ARP example



IP source: 192.0.2.2

IP destination: 192.0.2.3

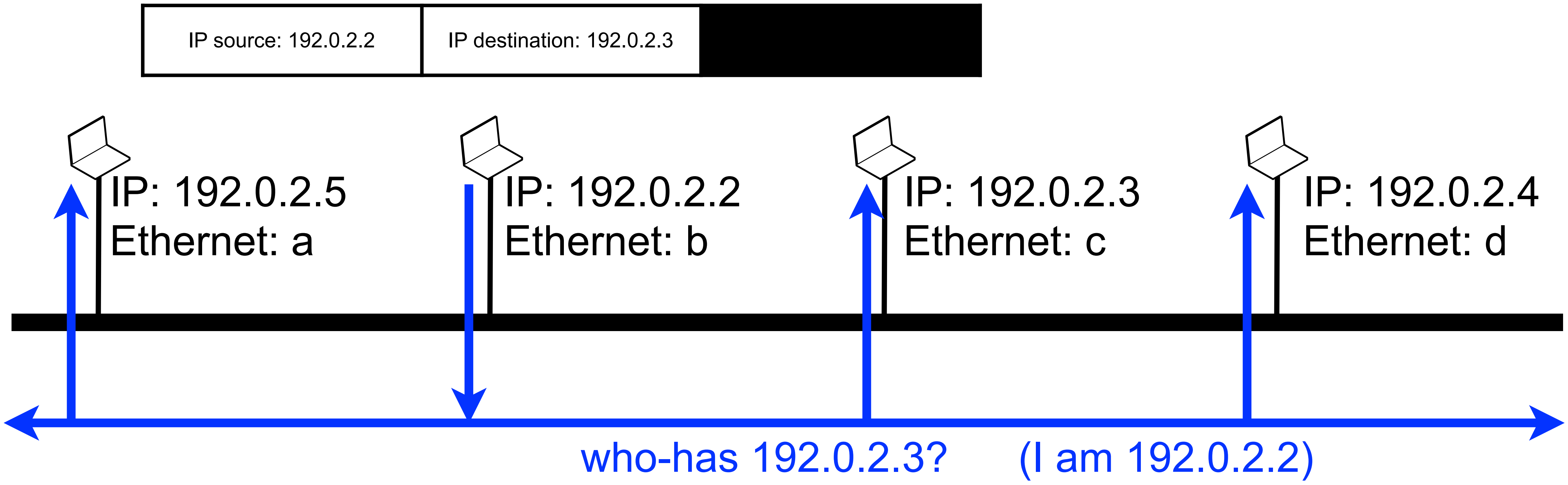
IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.3
Ethernet: c

IP: 192.0.2.4
Ethernet: d

ARP example



IP source: 192.0.2.2

IP destination: 192.0.2.3

IP: 192.0.2.5
Ethernet: a

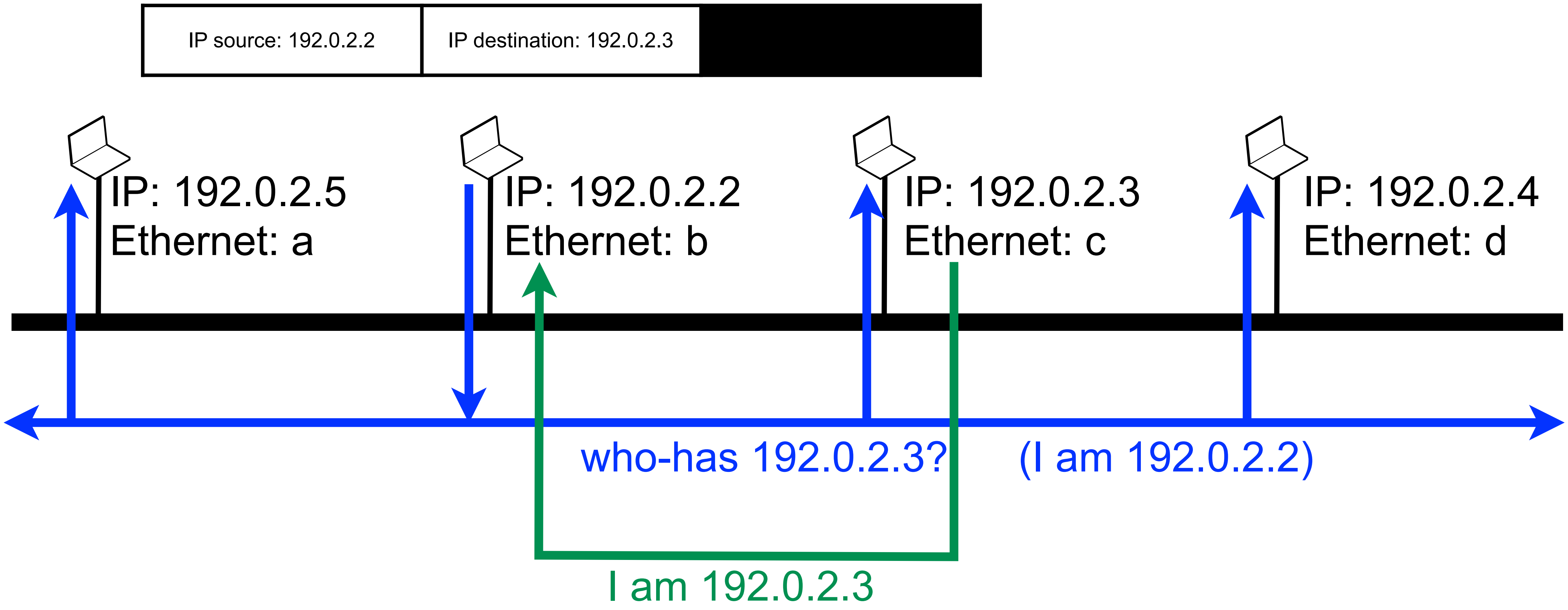
IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.3
Ethernet: c

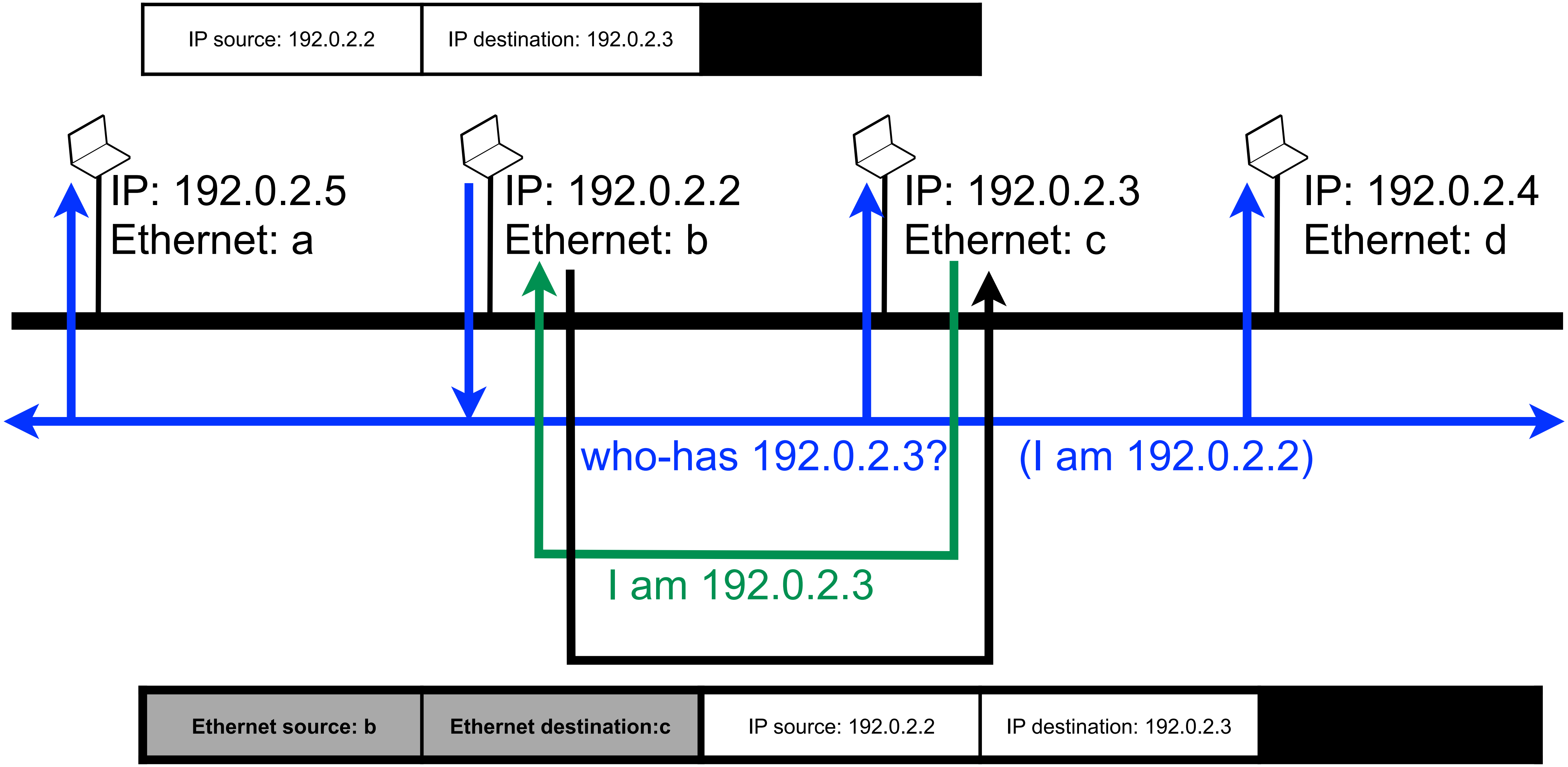
IP: 192.0.2.4
Ethernet: d

who-has 192.0.2.3? (I am 192.0.2.2)

ARP example


IP source: 192.0.2.2
IP destination: 192.0.2.3
IP: 192.0.2.5
Ethernet: a
IP: 192.0.2.2
Ethernet: b
IP: 192.0.2.3
Ethernet: c
IP: 192.0.2.4
Ethernet: d
who-has 192.0.2.3?
(I am 192.0.2.2)
I am 192.0.2.3

ARP example



IP source: 192.0.2.2	IP destination: 192.0.2.3	

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.3
Ethernet: c

IP: 192.0.2.4
Ethernet: d

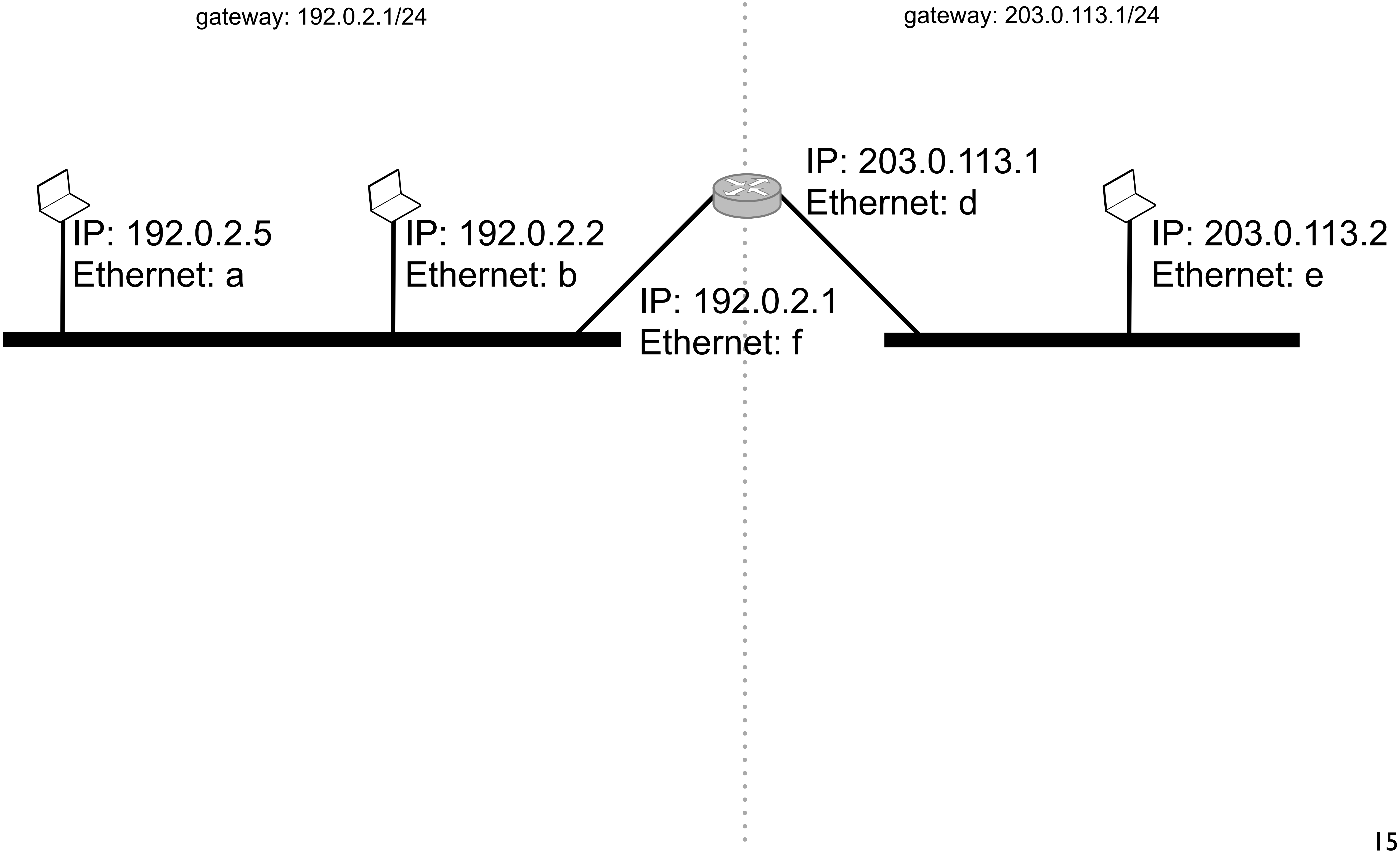
who-has 192.0.2.3?

(I am 192.0.2.2)

I am 192.0.2.3

Ethernet source: b	**Ethernet destination:c**	IP source: 192.0.2.2	IP destination: 192.0.2.3	

ARP example (router)



gateway: 192.0.2.1/24

gateway: 203.0.113.1/24

IP: 203.0.113.1
Ethernet: d

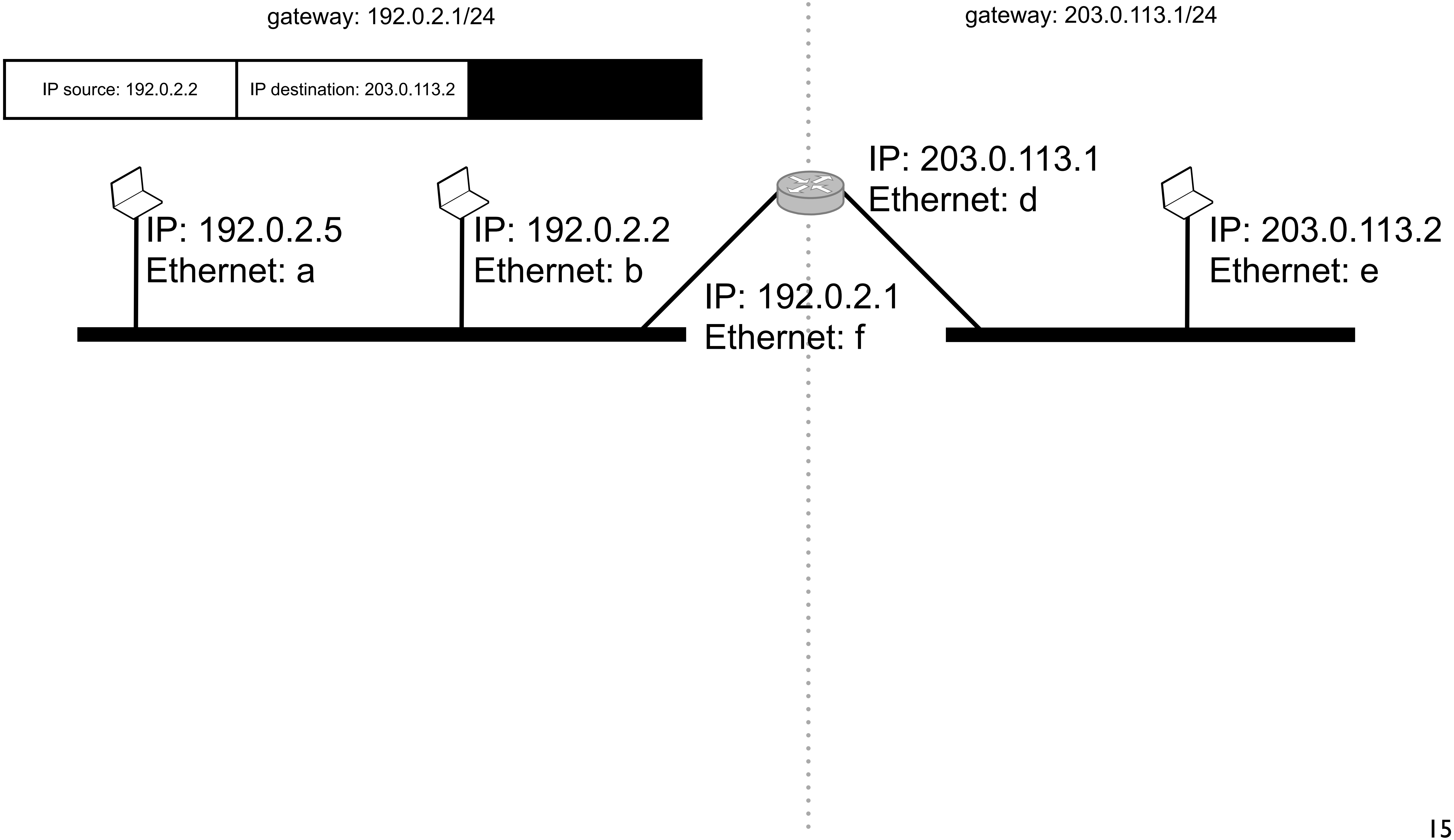
IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.1
Ethernet: f

IP: 203.0.113.2
Ethernet: e

ARP example (router)



gateway: 192.0.2.1/24

gateway: 203.0.113.1/24

IP source: 192.0.2.2

IP destination: 203.0.113.2

IP: 192.0.2.5
Ethernet: a

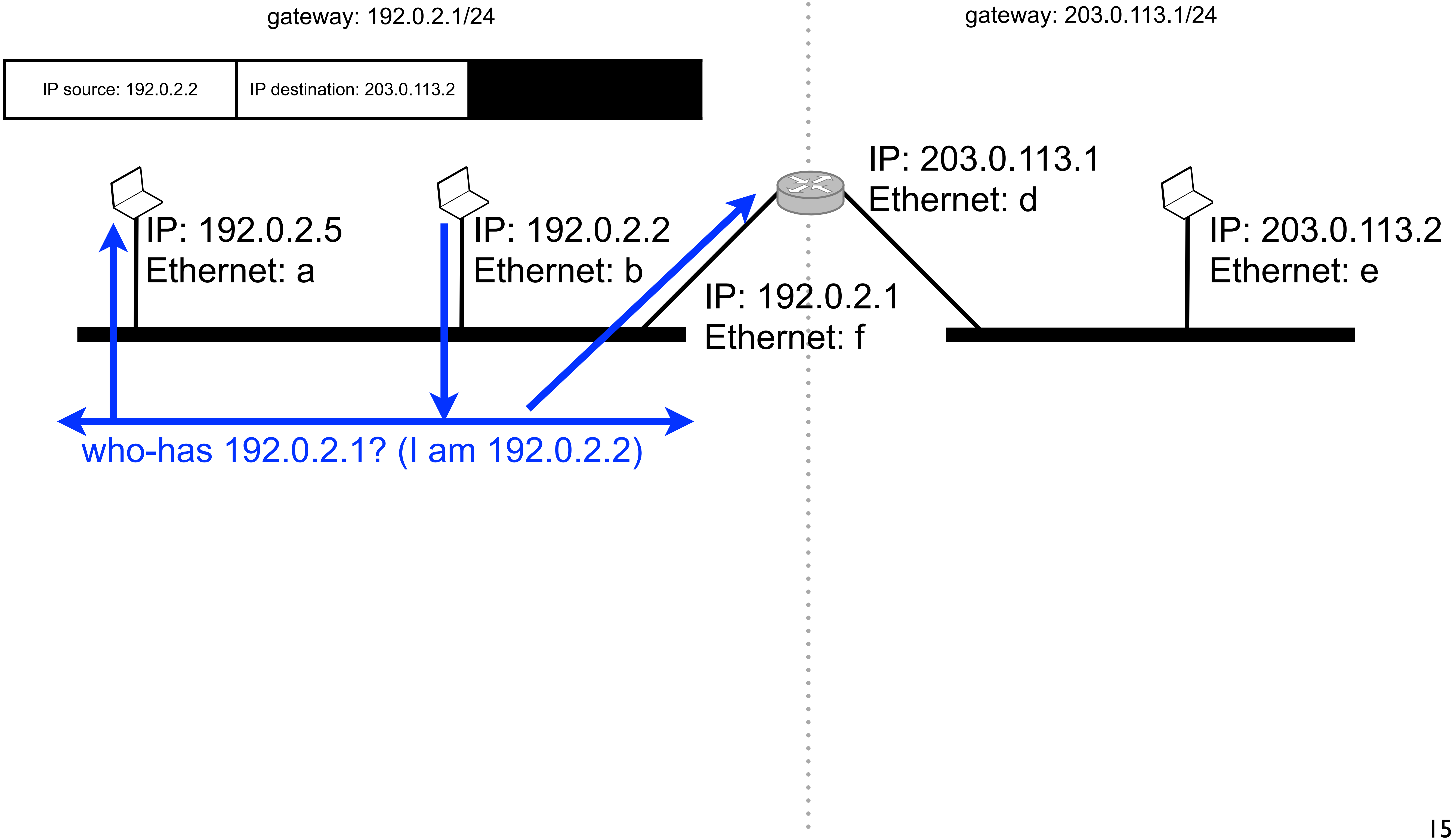
IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.1
Ethernet: f

IP: 203.0.113.1
Ethernet: d

IP: 203.0.113.2
Ethernet: e

ARP example (router)



gateway: 192.0.2.1/24

gateway: 203.0.113.1/24

IP source: 192.0.2.2

IP destination: 203.0.113.2

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

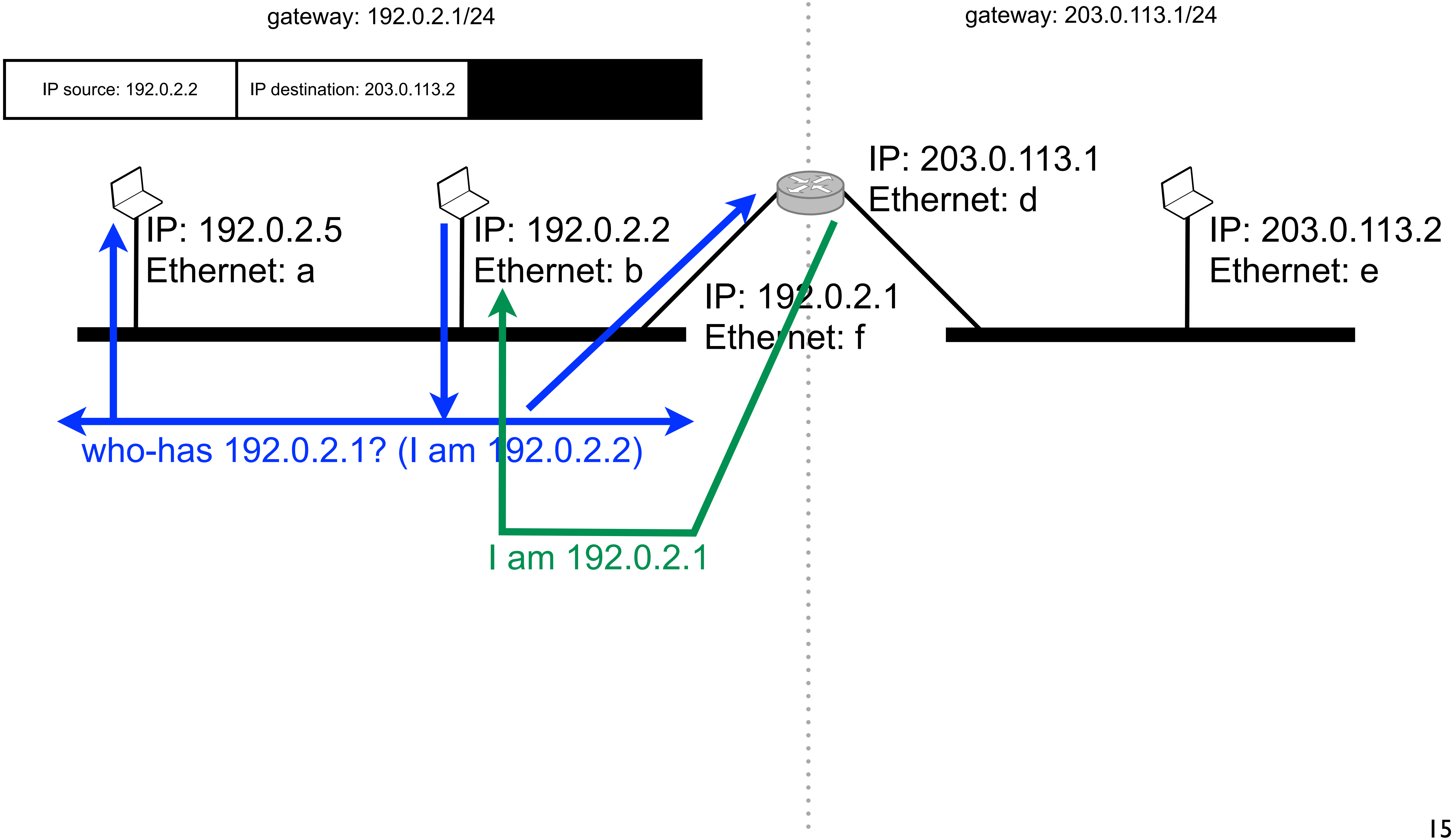
IP: 192.0.2.1
Ethernet: f

IP: 203.0.113.1
Ethernet: d

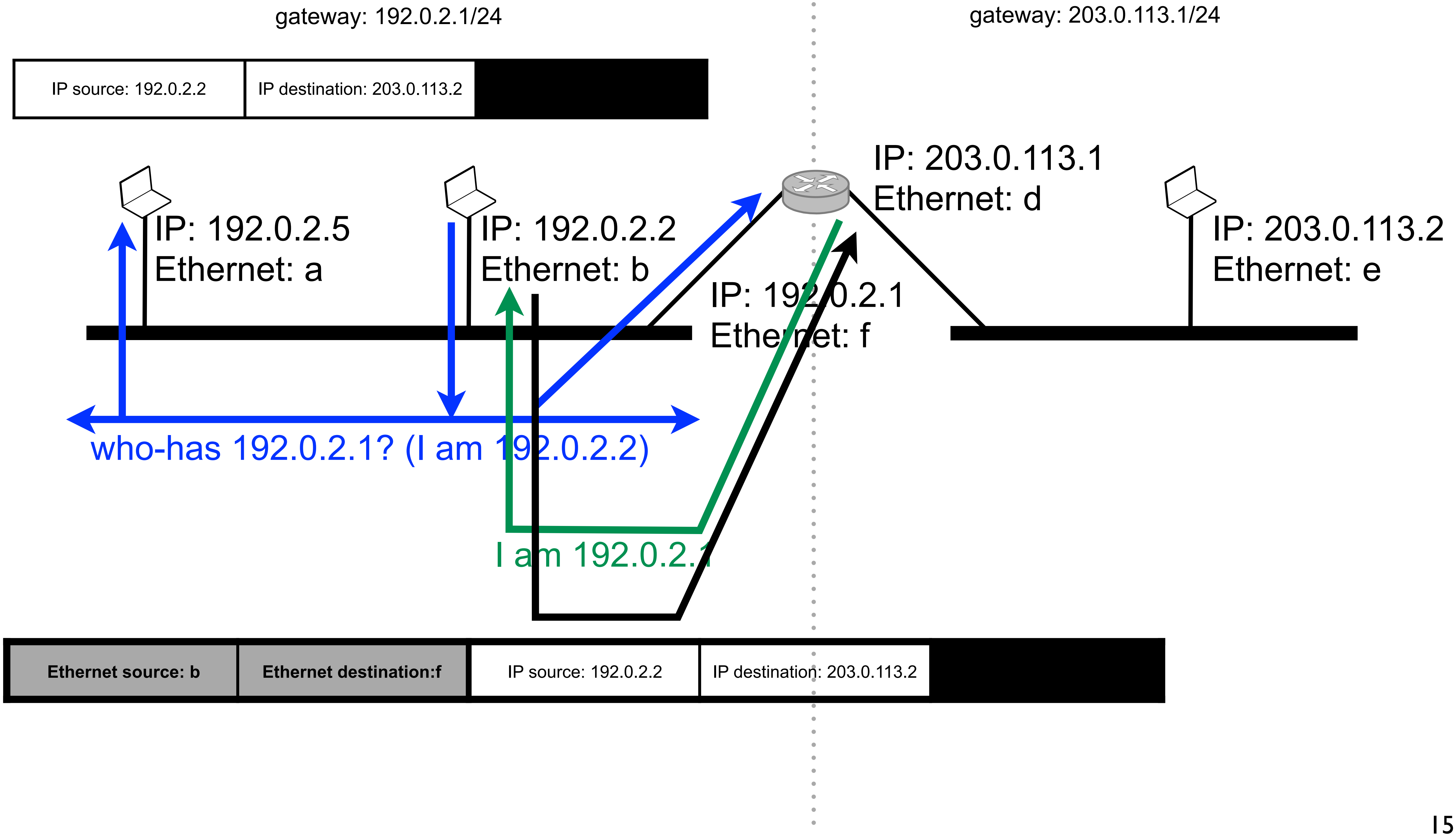
IP: 203.0.113.2
Ethernet: e

who-has 192.0.2.1? (I am 192.0.2.2)

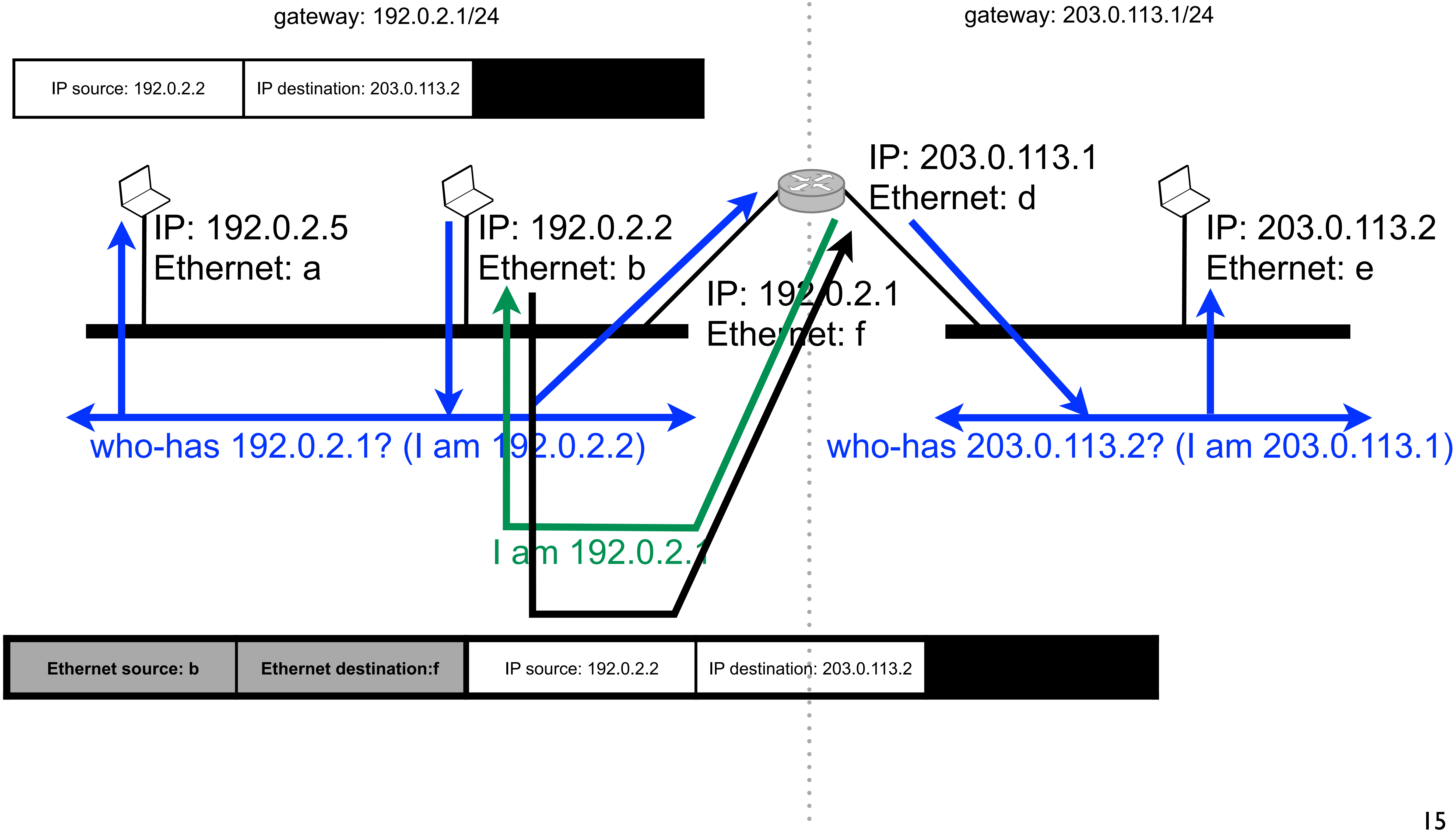
ARP example (router)



ARP example (router)



ARP example (router)



gateway: 192.0.2.1/24

gateway: 203.0.113.1/24

IP source: 192.0.2.2	IP destination: 203.0.113.2

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 203.0.113.1
Ethernet: d

IP: 192.0.2.1
Ethernet: f

IP: 203.0.113.2
Ethernet: e

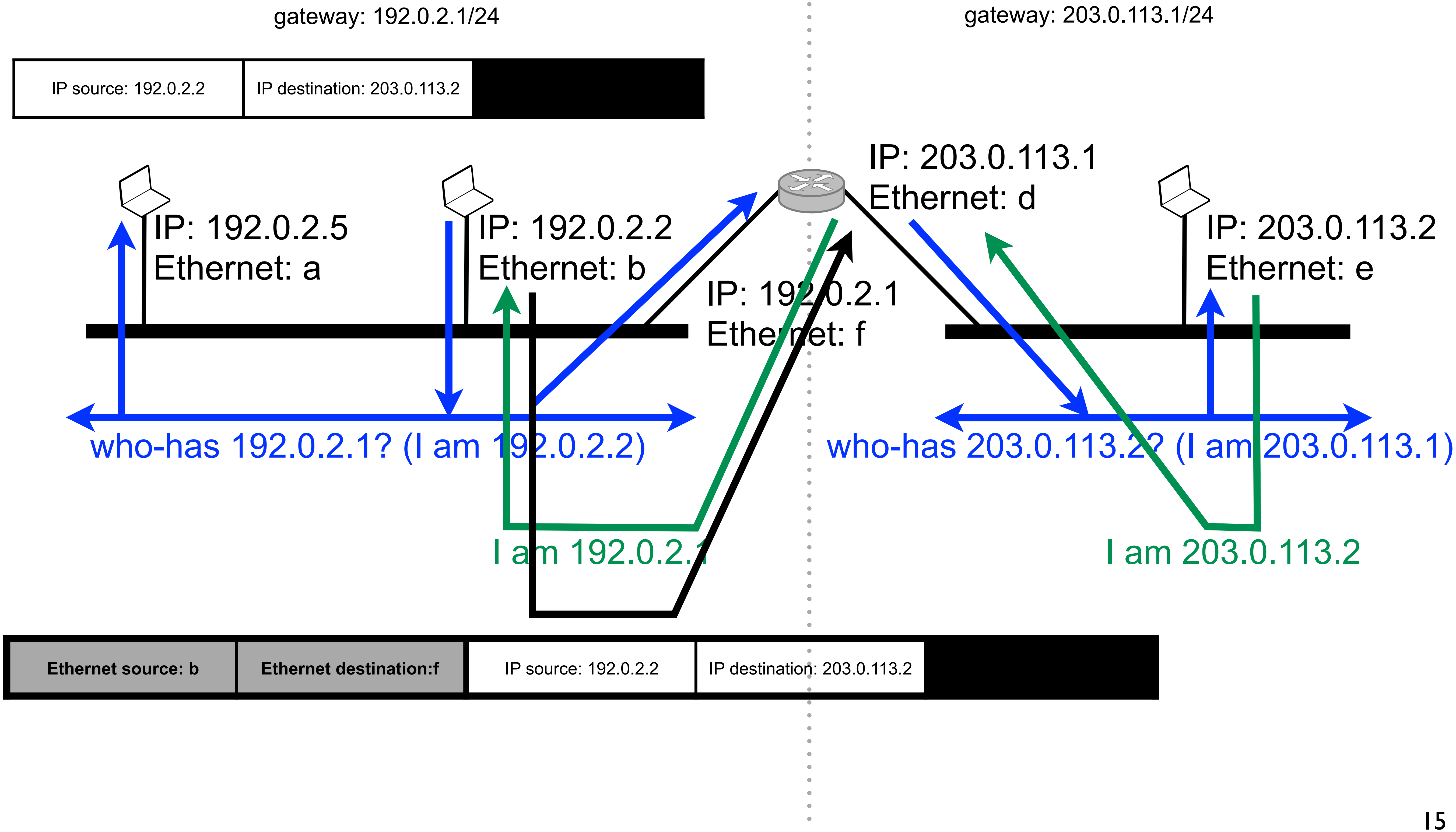
who-has 192.0.2.1? (I am 192.0.2.2)

who-has 203.0.113.2? (I am 203.0.113.1)

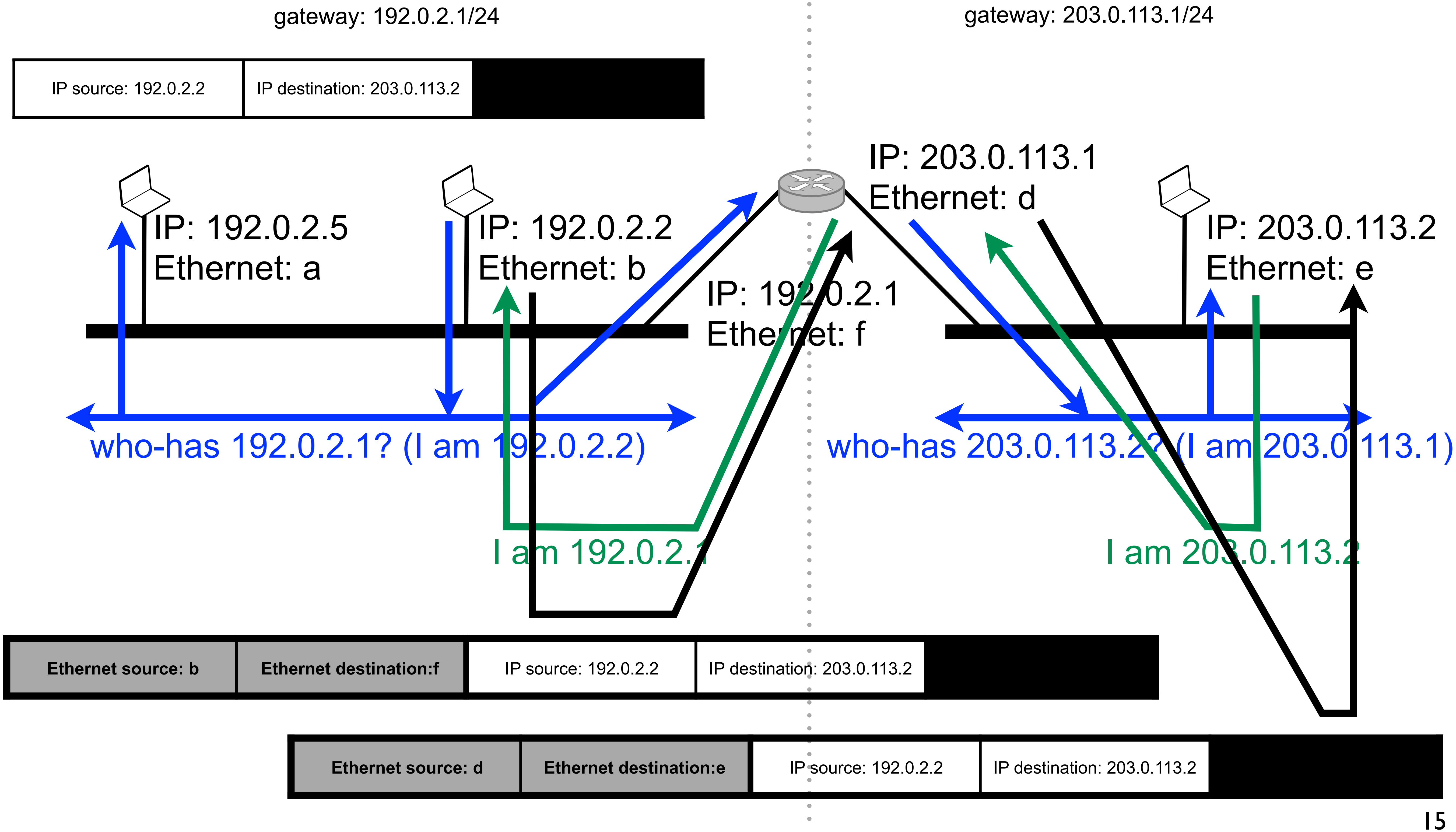
I am 192.0.2.1

Ethernet source: b	Ethernet destination:f	IP source: 192.0.2.2	IP destination: 203.0.113.2

ARP example (router)



ARP example (router)



gateway: 192.0.2.1/24

gateway: 203.0.113.1/24

IP source: 192.0.2.2	IP destination: 203.0.113.2	

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.1
Ethernet: f

IP: 203.0.113.1
Ethernet: d

IP: 203.0.113.2
Ethernet: e

who-has 192.0.2.1? (I am 192.0.2.2)

I am 192.0.2.1

who-has 203.0.113.2? (I am 203.0.113.1)

I am 203.0.113.2

Ethernet source: b	Ethernet destination:f	IP source: 192.0.2.2	IP destination: 203.0.113.2	

Ethernet source: d	Ethernet destination:e	IP source: 192.0.2.2	IP destination: 203.0.113.2	

Dynamic address configuration

- Allow a set of hosts to share a pool of IP address
- Two approaches
  - stateless auto-configuration
    - no infrastructure necessary
  - Dynamic Host Configuration Protocol (DHCP)
    - hosts query a DHCP server to obtain their configuration
- Advantages
  - less address wastage: a host can use the address of another hosts when it is not connected
  - improves flexibility and reduces the risk of configuration error as no manual operation is necessary

Stateless auto-configuration

- When a host connects to the network:

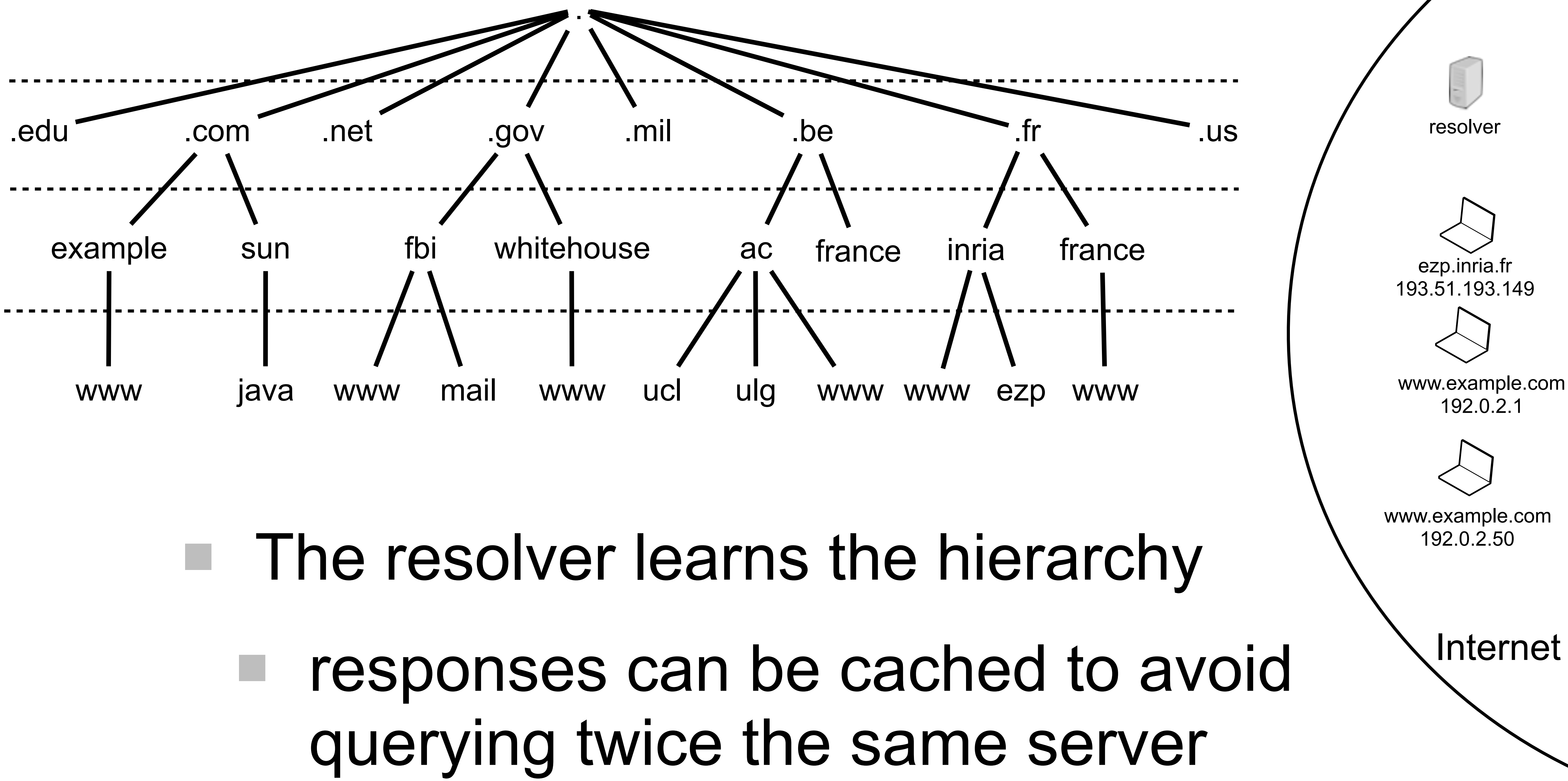
1. The host choses an address randomly in 169.254/16 (not globally routable)
2. Sends an ARP request for the chosen address
3. If an ARP reply is received (another host already uses the address
   - restart from point 1
4. Otherwise, the address the address is not used by another host and the host can use it safely

- Auto-configuration is used only for communications within the same network
  - In IPv6, hosts can auto-configure their globally routable addresses and discover network services (e.g., routers, DNS...)

Dynamic Host Configuration Protocol (DHCP)

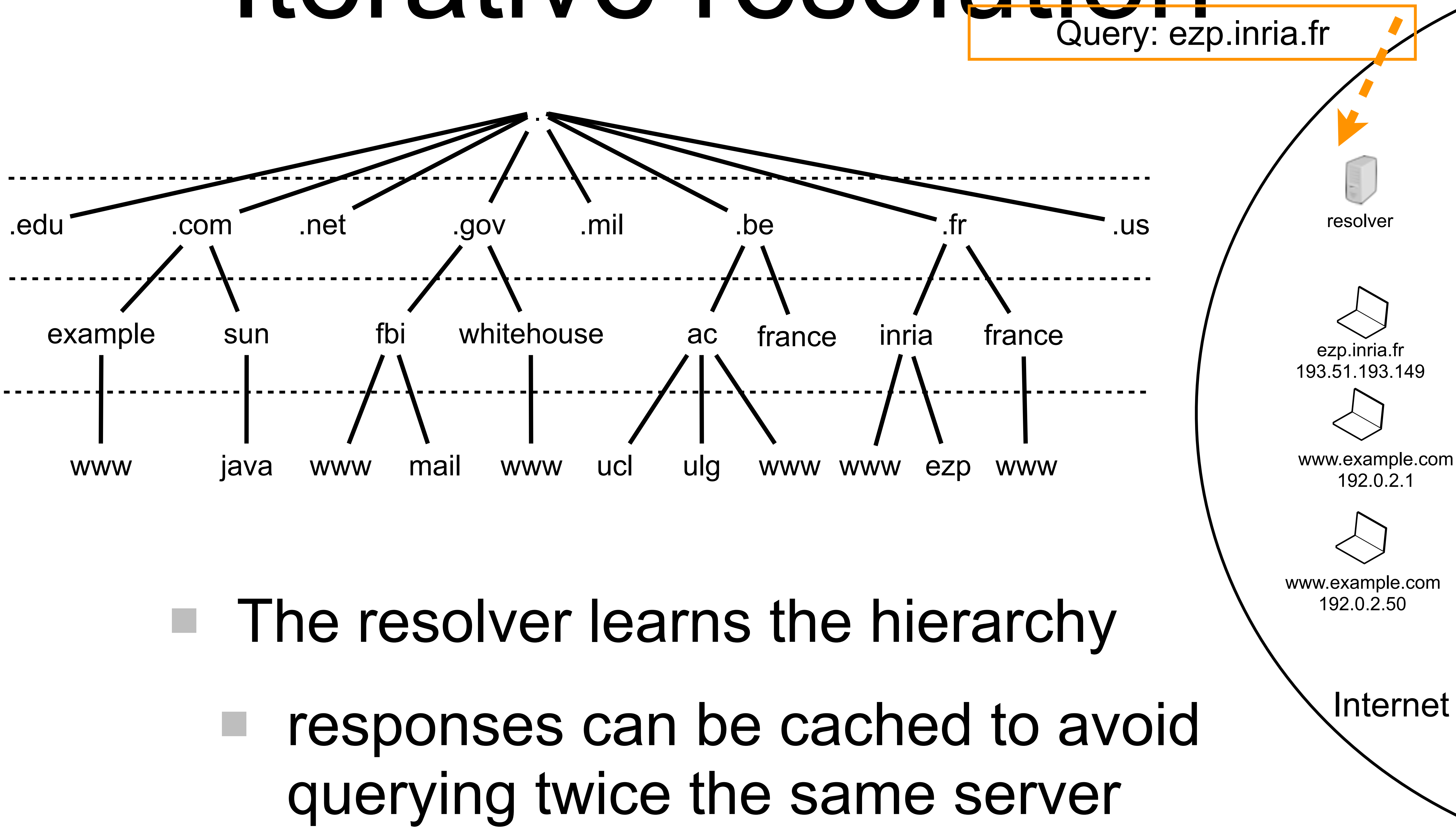
- When a host connects to the network, it broadcasts a DHCP discovery datagram
- Any DHCP server that receives such a message replies with a DHCP offer datagram that contains an offer of IP address
- The host picks one offer and broadcasts a DHCP request message to announce the offers it selected
- The selected DHCP server assigns the address to the host and sends it back a DHCP acknowledgment that confirms the lease of the address and give additional parameters such as the lease time, the IP address of the default gateway, or the IP address of the DNS servers
    - when the lease time is elapsed, the address is released and made available for other hosts
- The other DHCP servers withdraw their offers

Iterative resolution



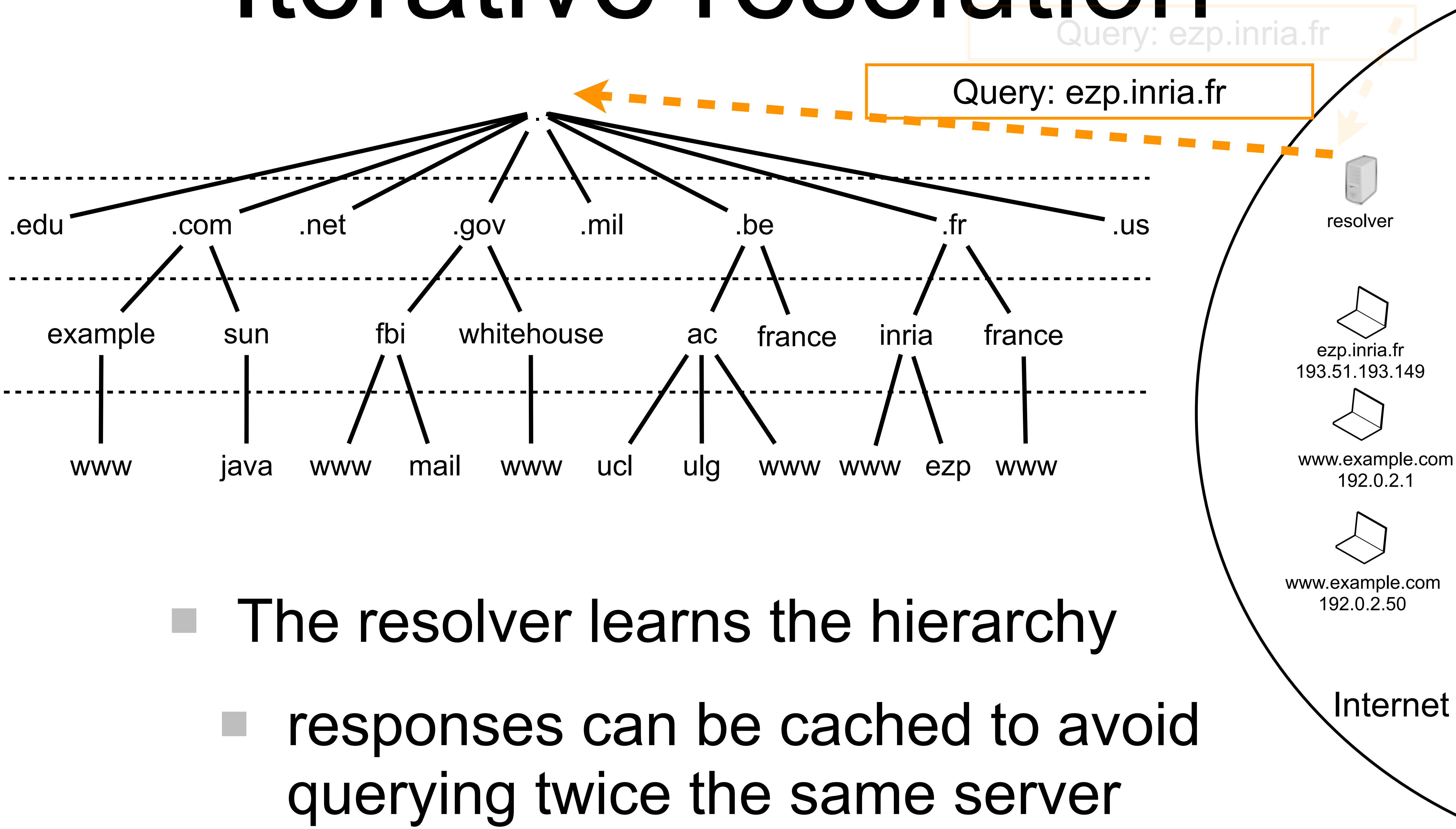
- The resolver learns the hierarchy
  - responses can be cached to avoid querying twice the same server

Iterative resolution



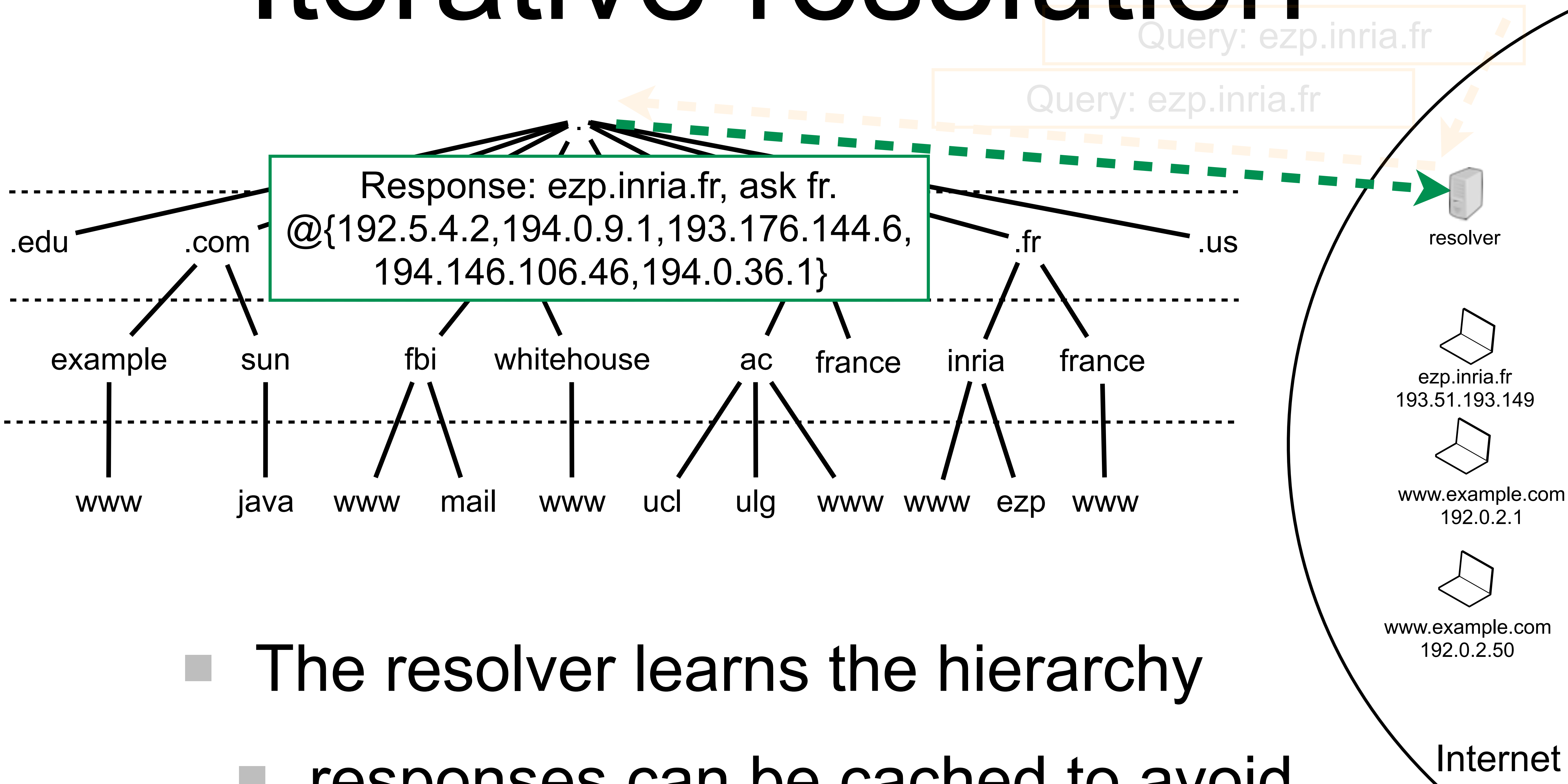
- The resolver learns the hierarchy
  - responses can be cached to avoid querying twice the same server

Iterative resolution



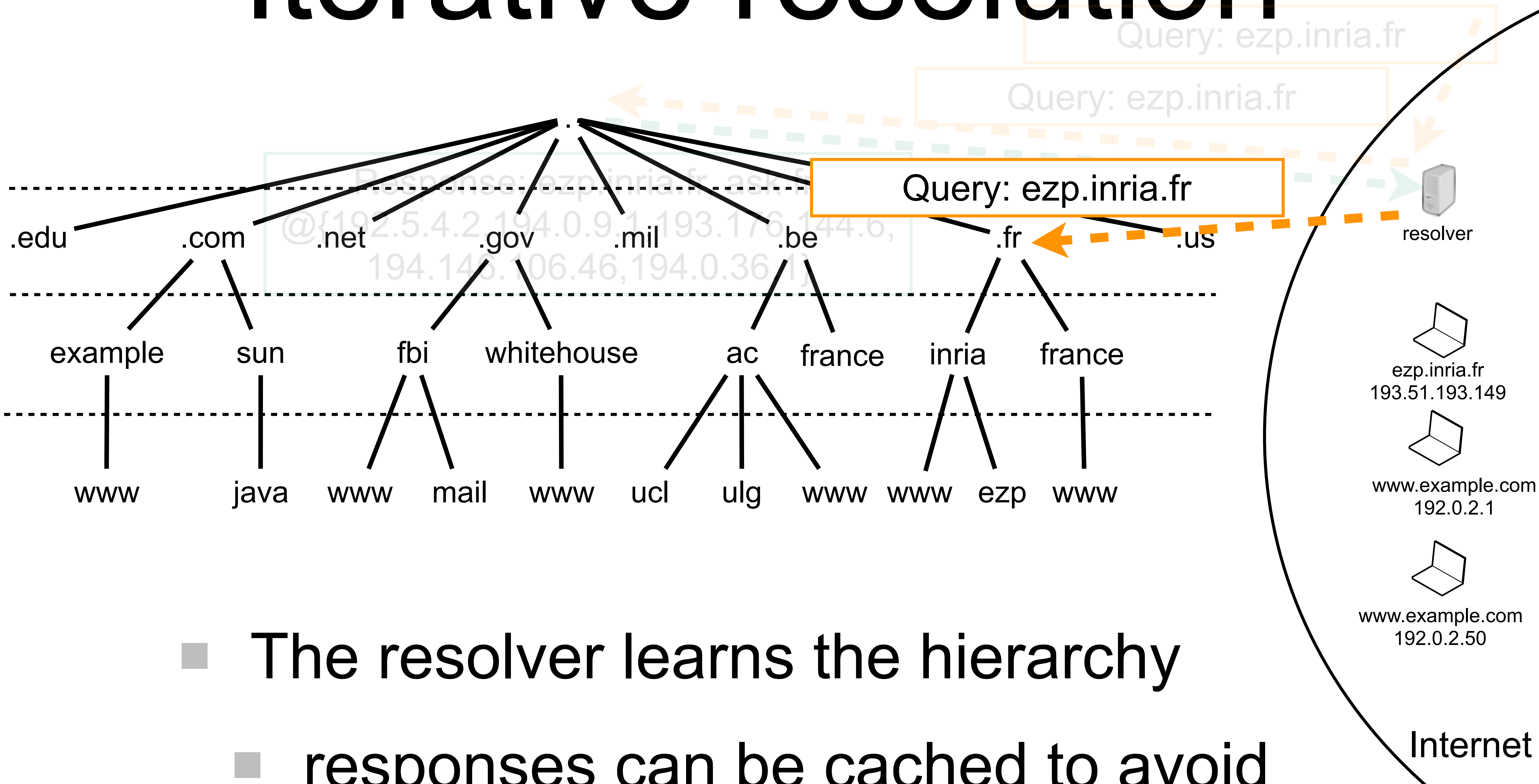
- The resolver learns the hierarchy
  - responses can be cached to avoid querying twice the same server

Iterative resolution



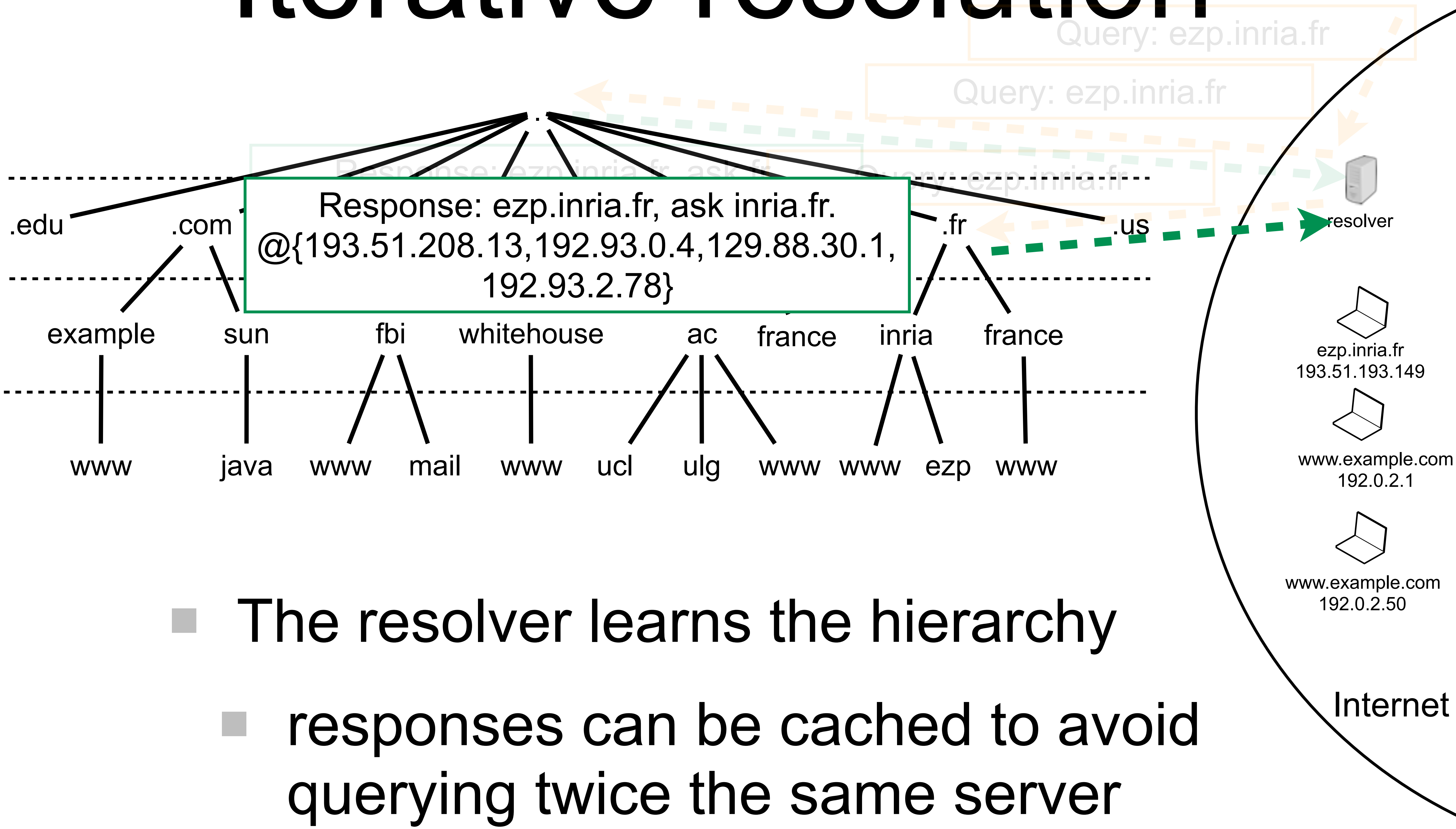
- The resolver learns the hierarchy
  - responses can be cached to avoid querying twice the same server

Iterative resolution



- The resolver learns the hierarchy
  - responses can be cached to avoid querying twice the same server

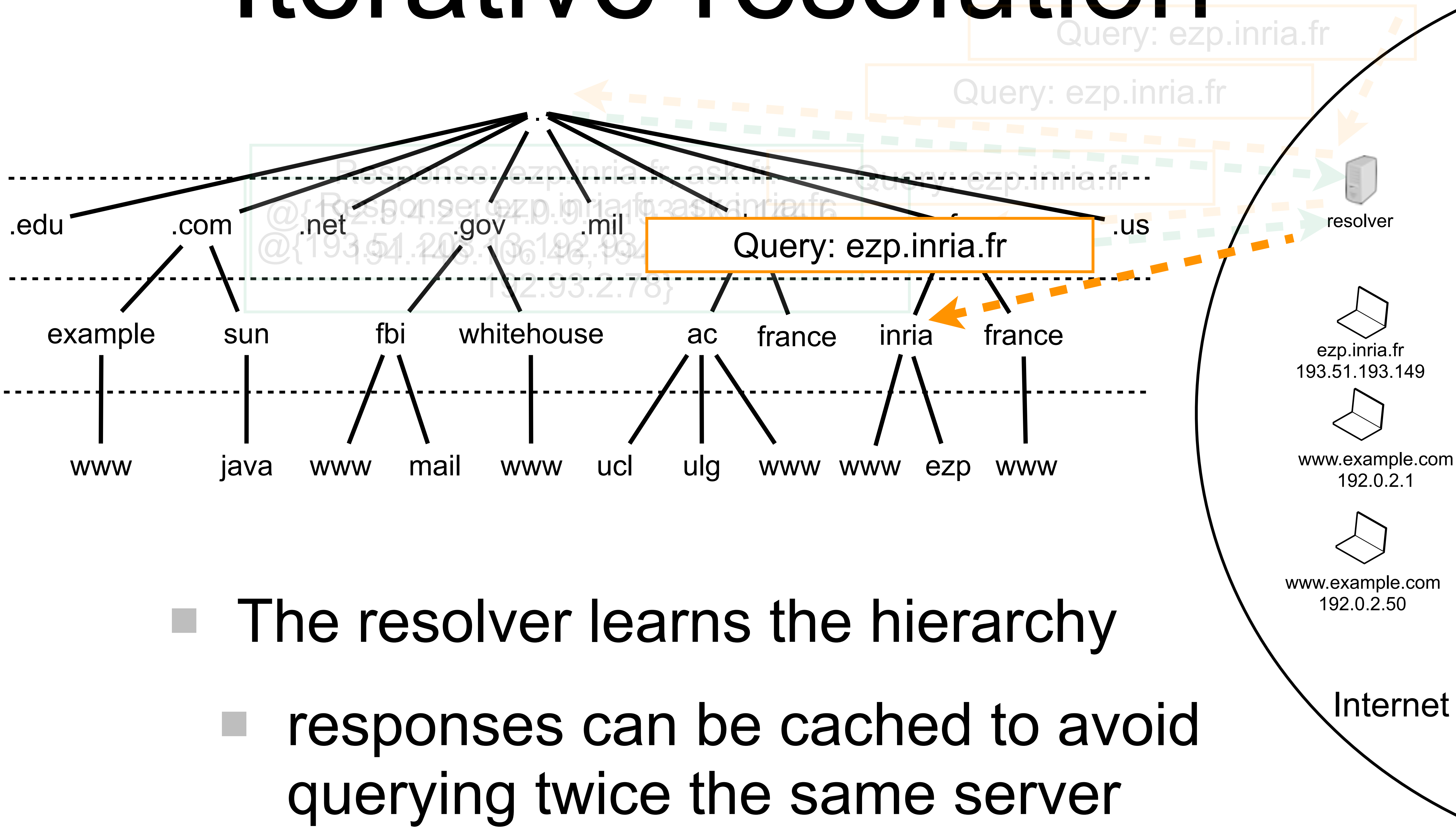
Iterative resolution



Response: ezp.inria.fr, ask inria.fr. @{193.51.208.13,192.93.0.4,129.88.30.1, 192.93.2.78}

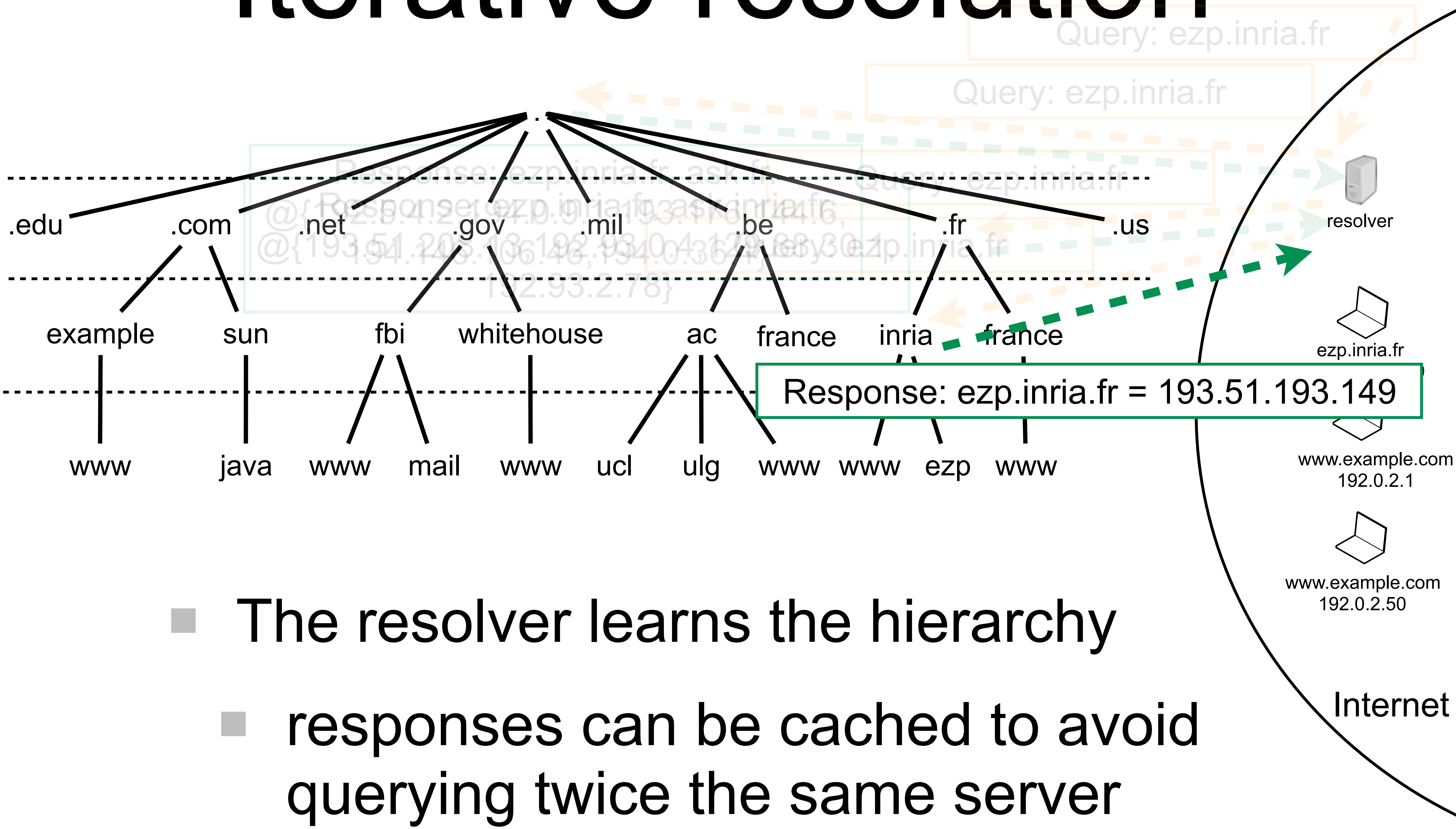
- The resolver learns the hierarchy
  - responses can be cached to avoid querying twice the same server

Iterative resolution



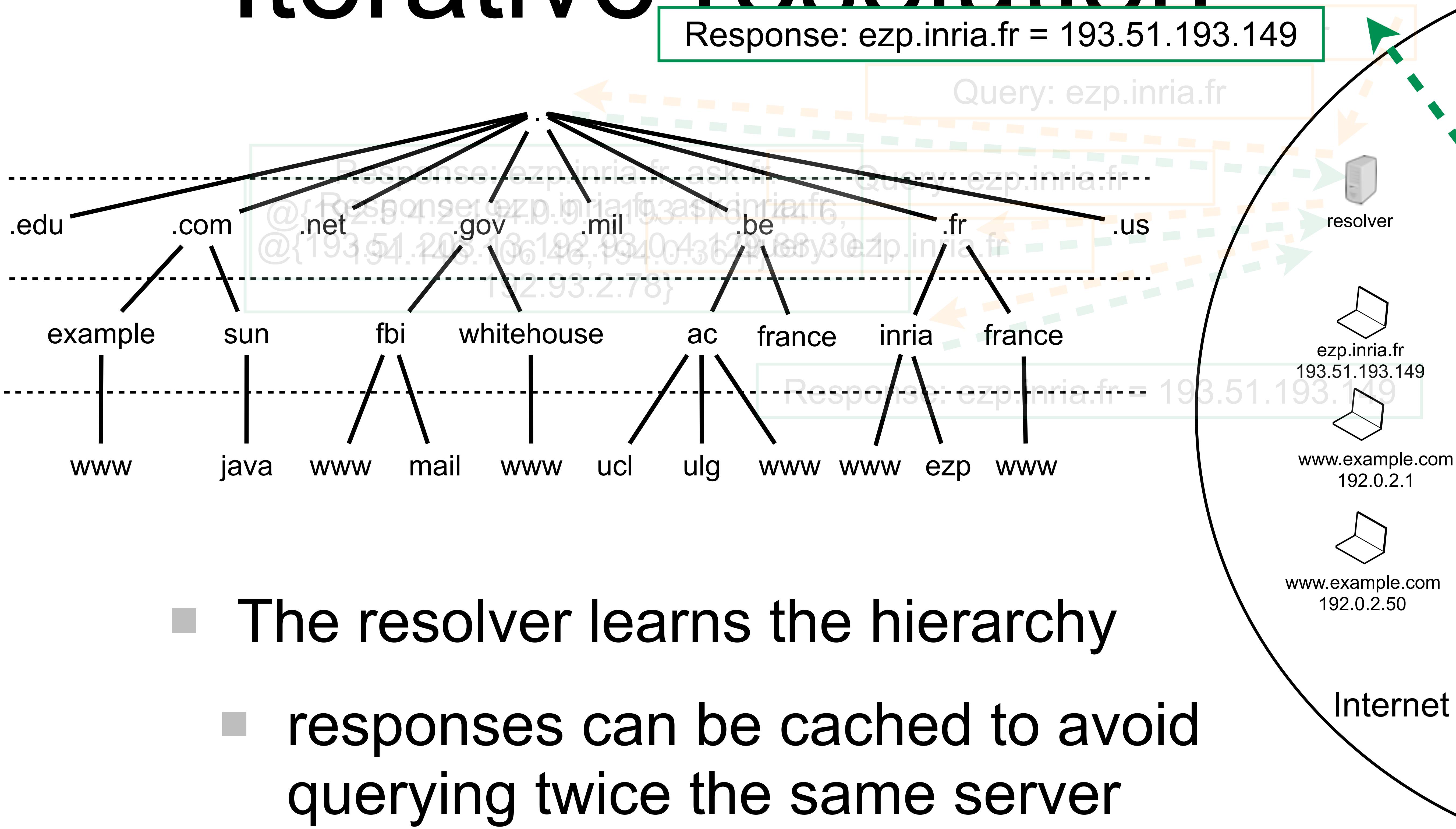
- The resolver learns the hierarchy
  - responses can be cached to avoid querying twice the same server

Iterative resolution



- The resolver learns the hierarchy
  - responses can be cached to avoid querying twice the same server

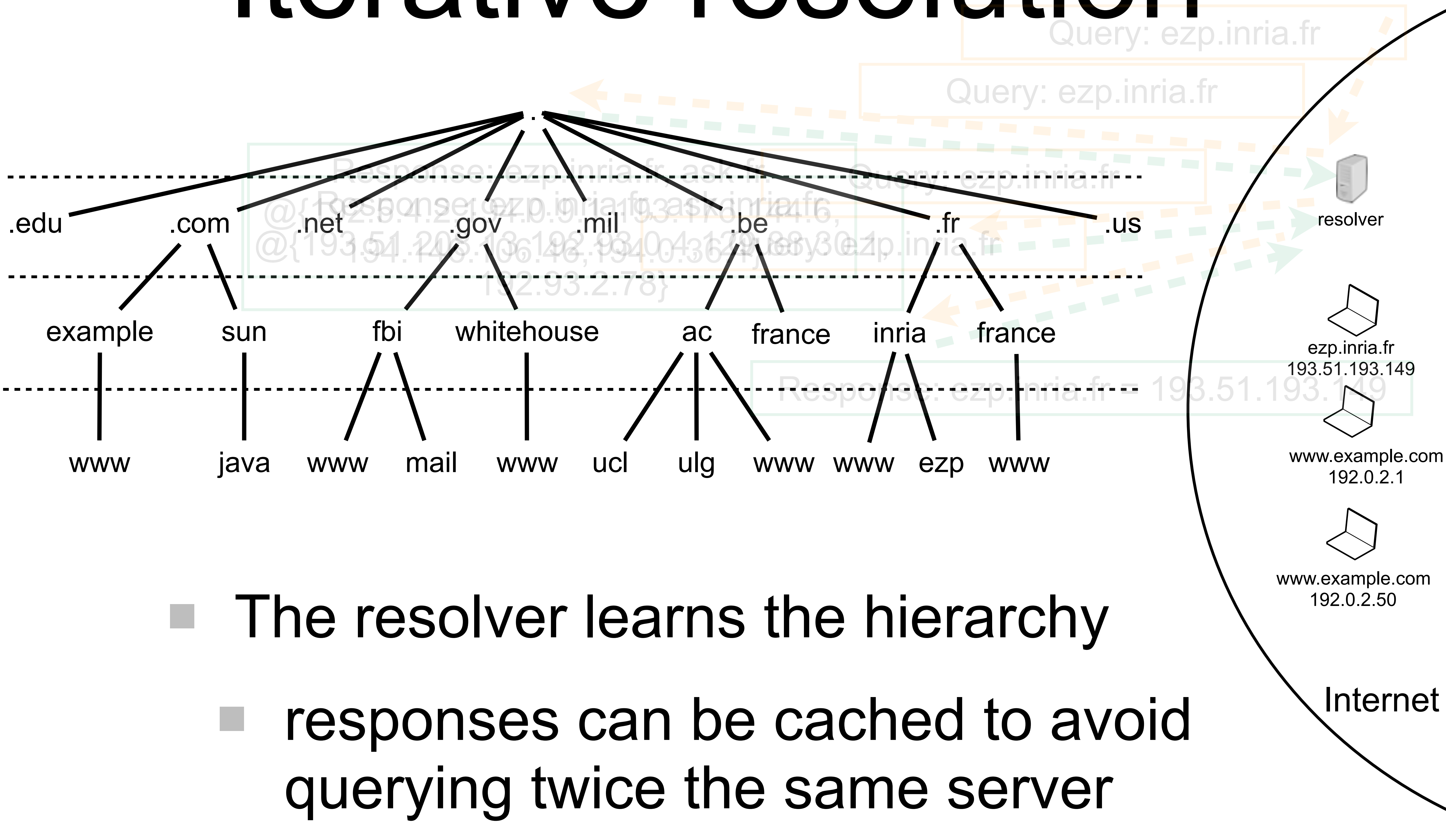
Iterative resolution



Response: ezp.inria.fr = 193.51.193.149

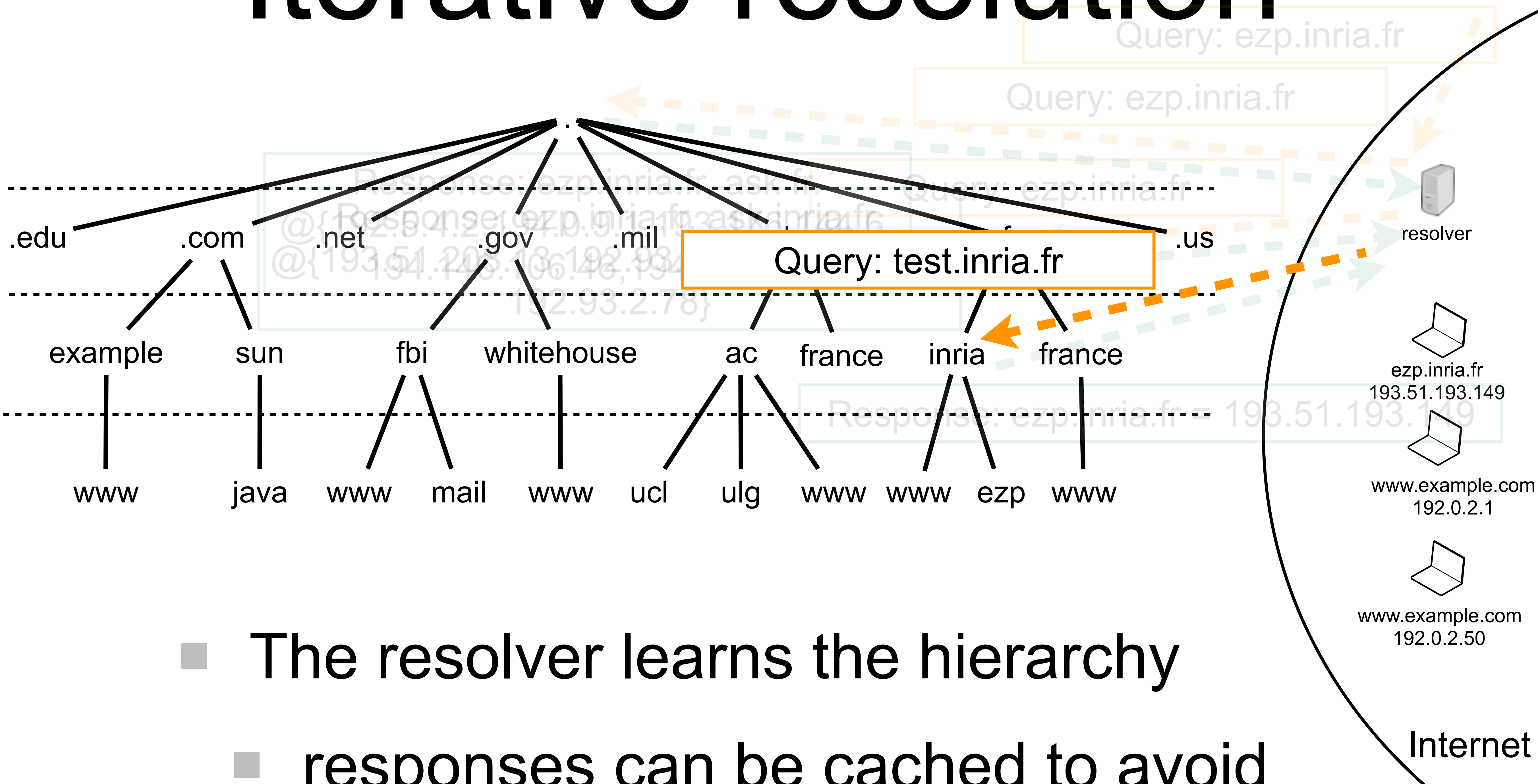
- The resolver learns the hierarchy
  - responses can be cached to avoid querying twice the same server

Iterative resolution



- The resolver learns the hierarchy
  - responses can be cached to avoid querying twice the same server

Iterative resolution



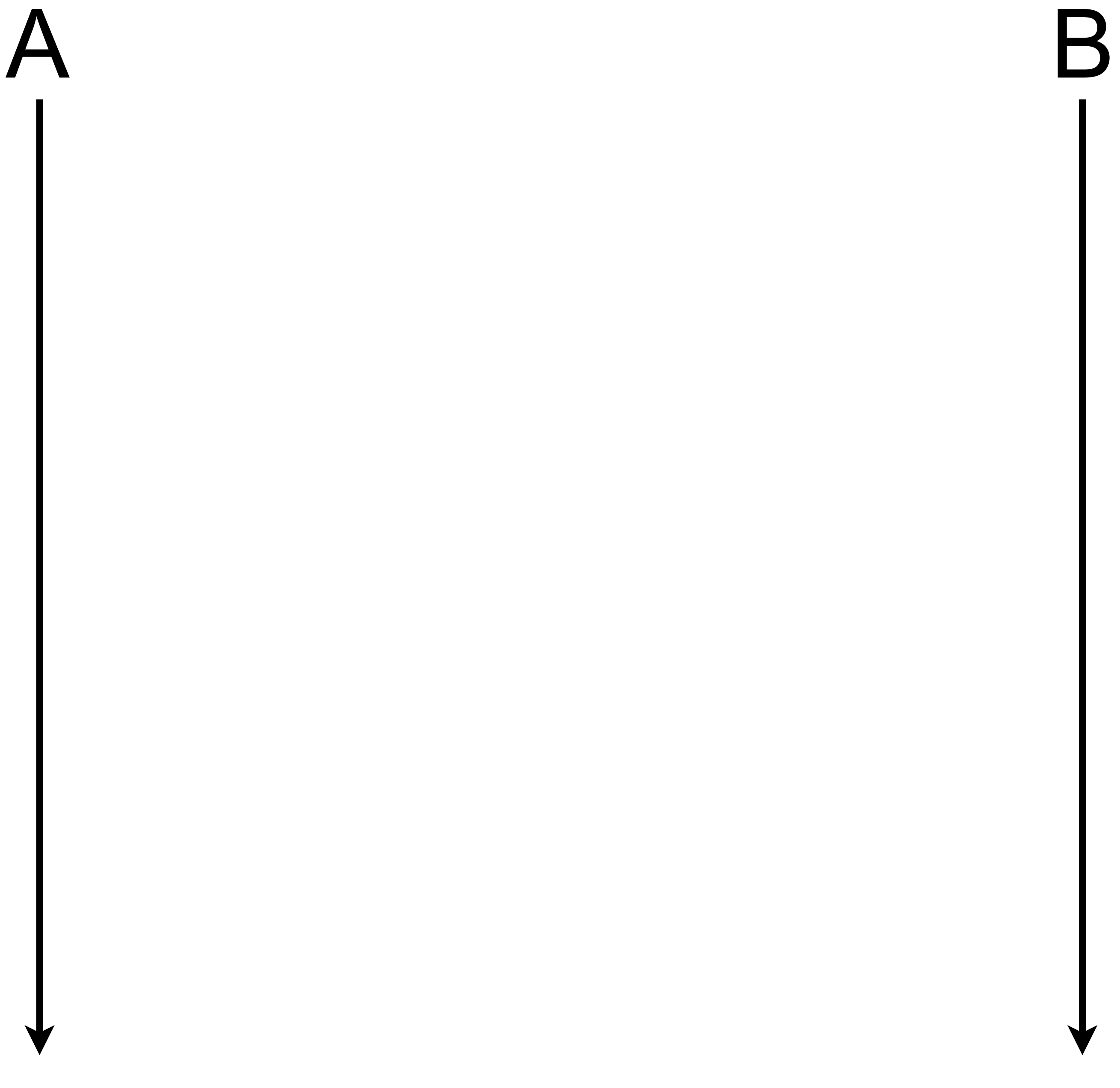
- The resolver learns the hierarchy
  - responses can be cached to avoid querying twice the same server

Transport

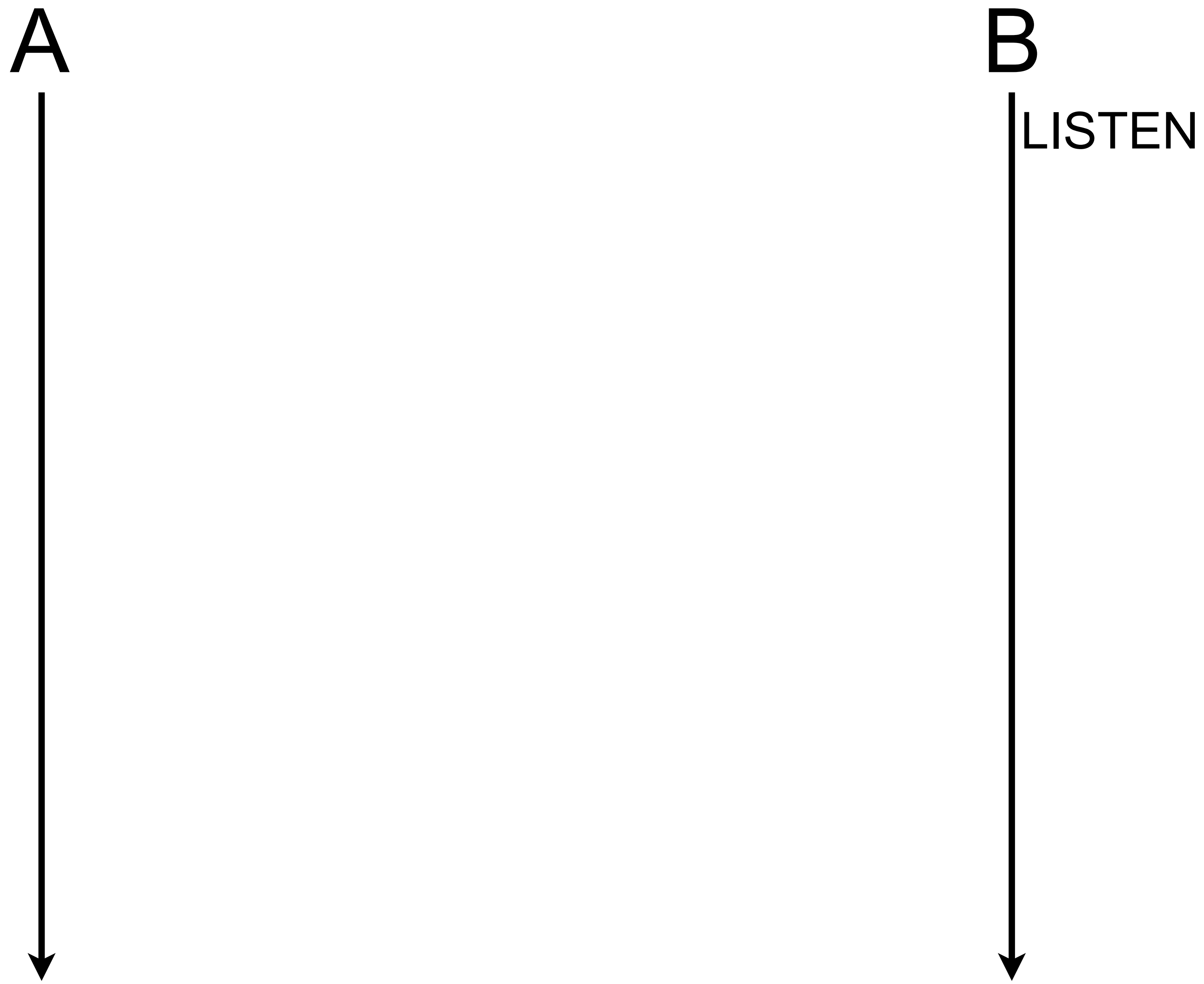
Transport of data between hosts

- Transport layer provides an end-to-end communication service
  - applications just deal with stream of bytes
- Most popular protocols:
  - UDP: connection-less, non reliable
  - TCP: connection-full, reliable

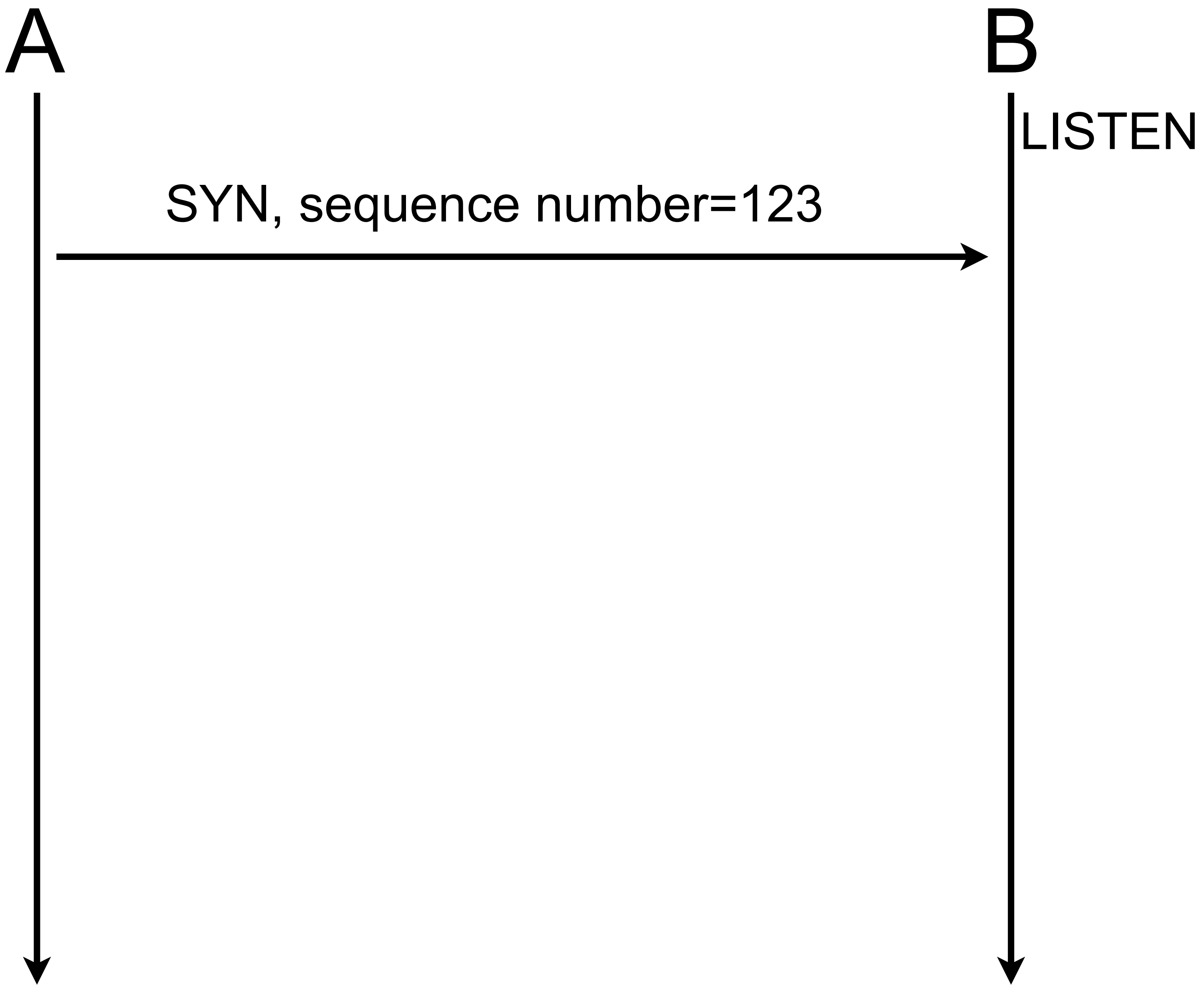
TCP connection establishment



TCP connection establishment



TCP connection establishment



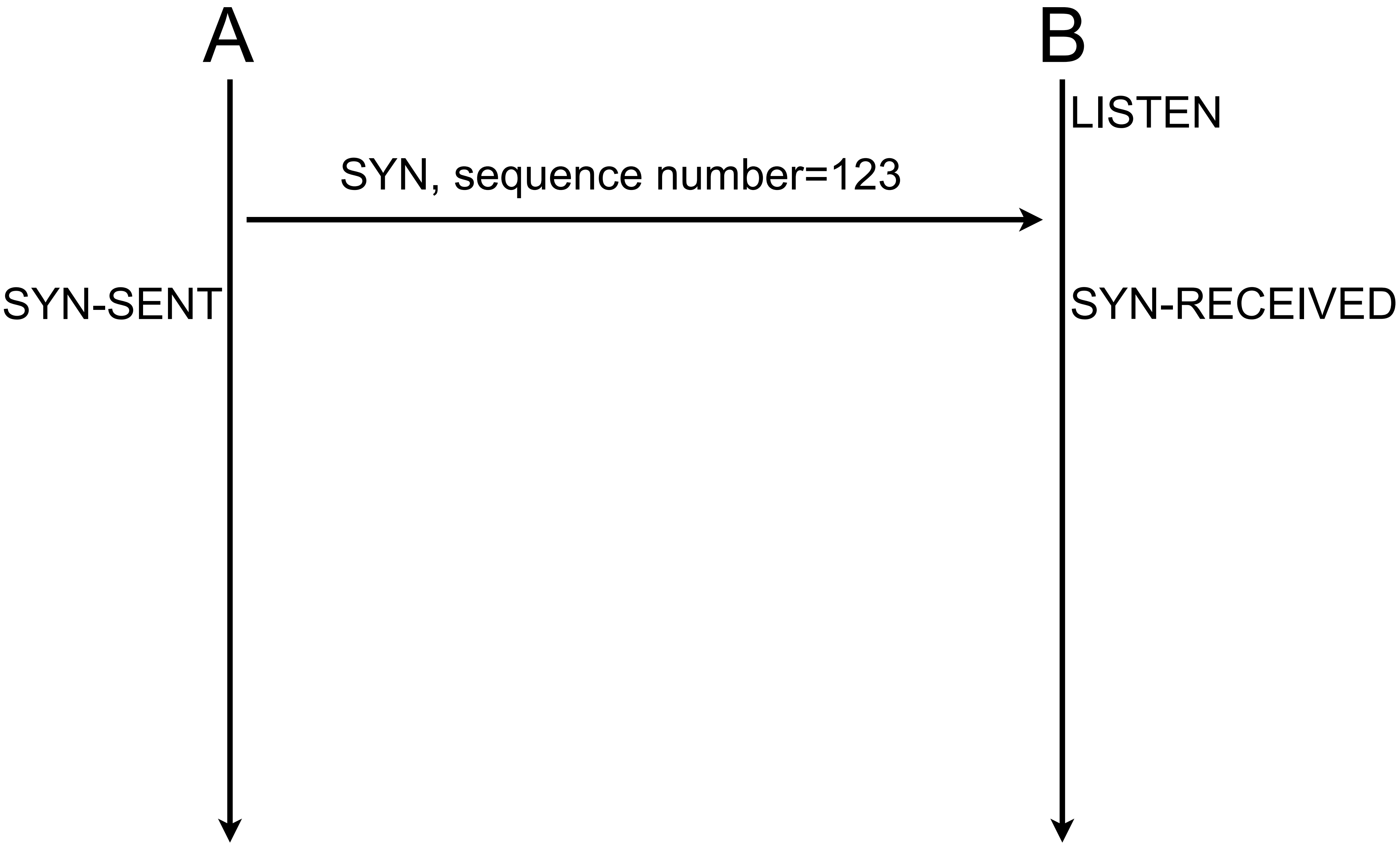
A

B

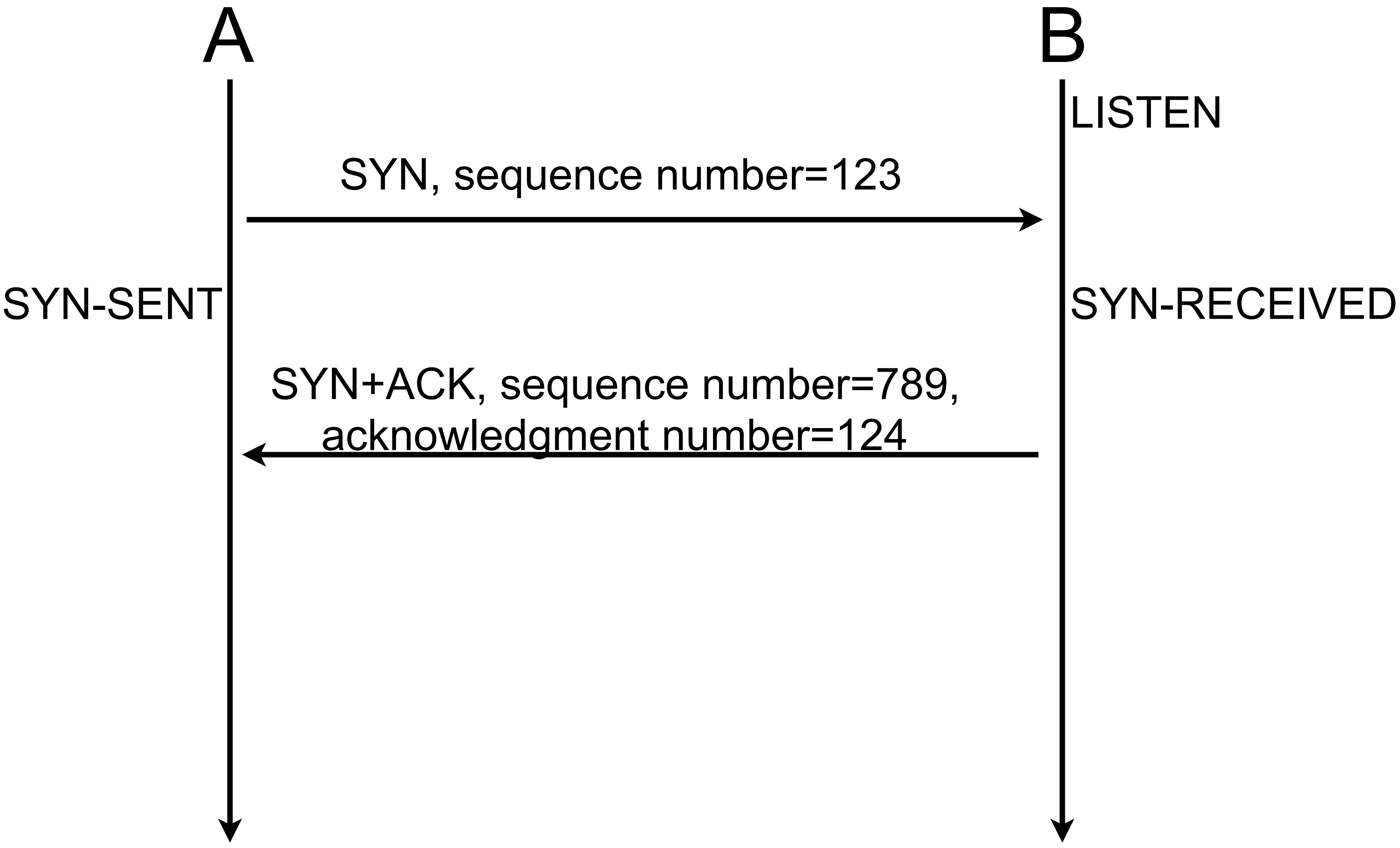
LISTEN

SYN, sequence number=123

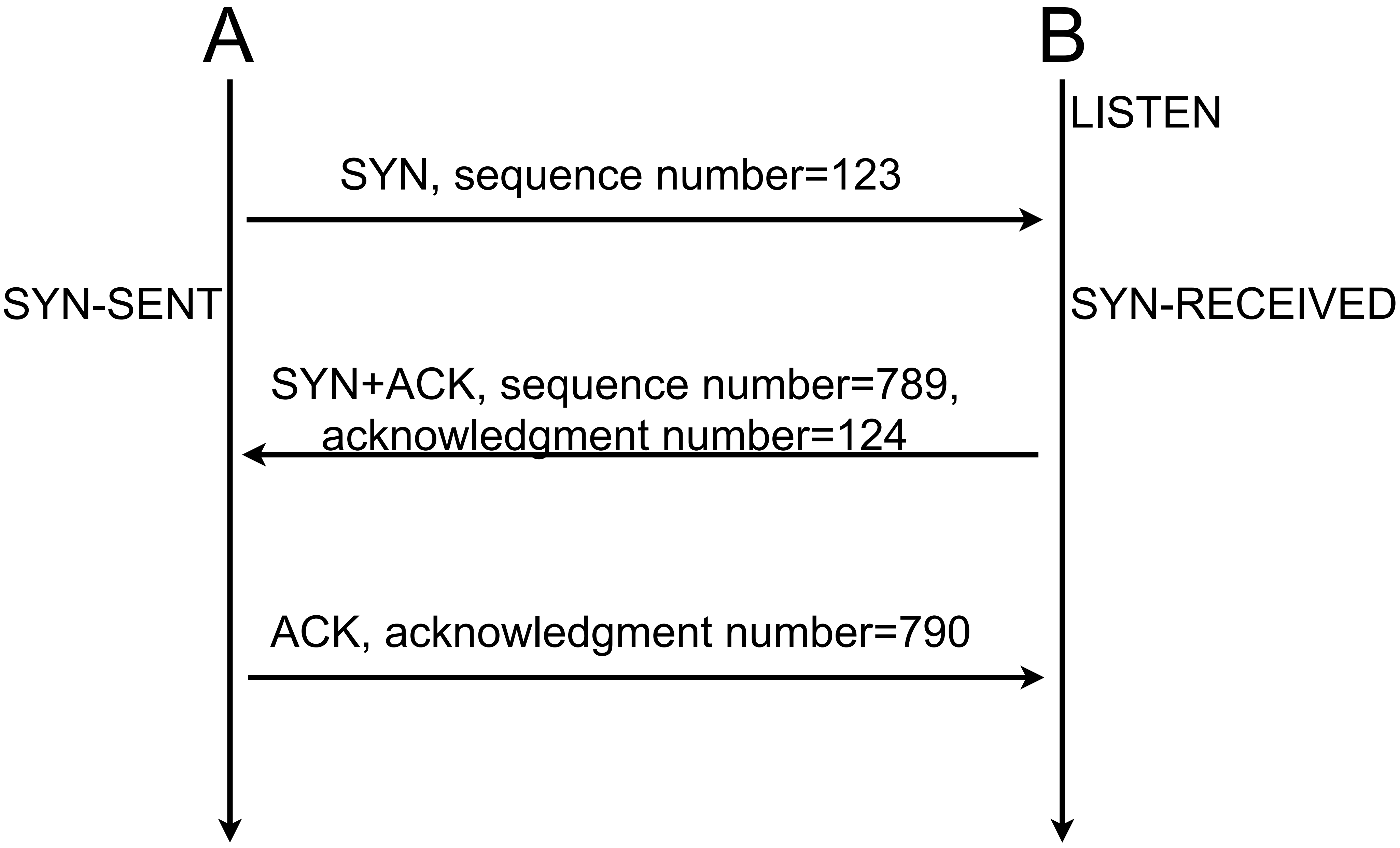
TCP connection establishment



TCP connection establishment



TCP connection establishment



A

B

LISTEN

SYN, sequence number=123

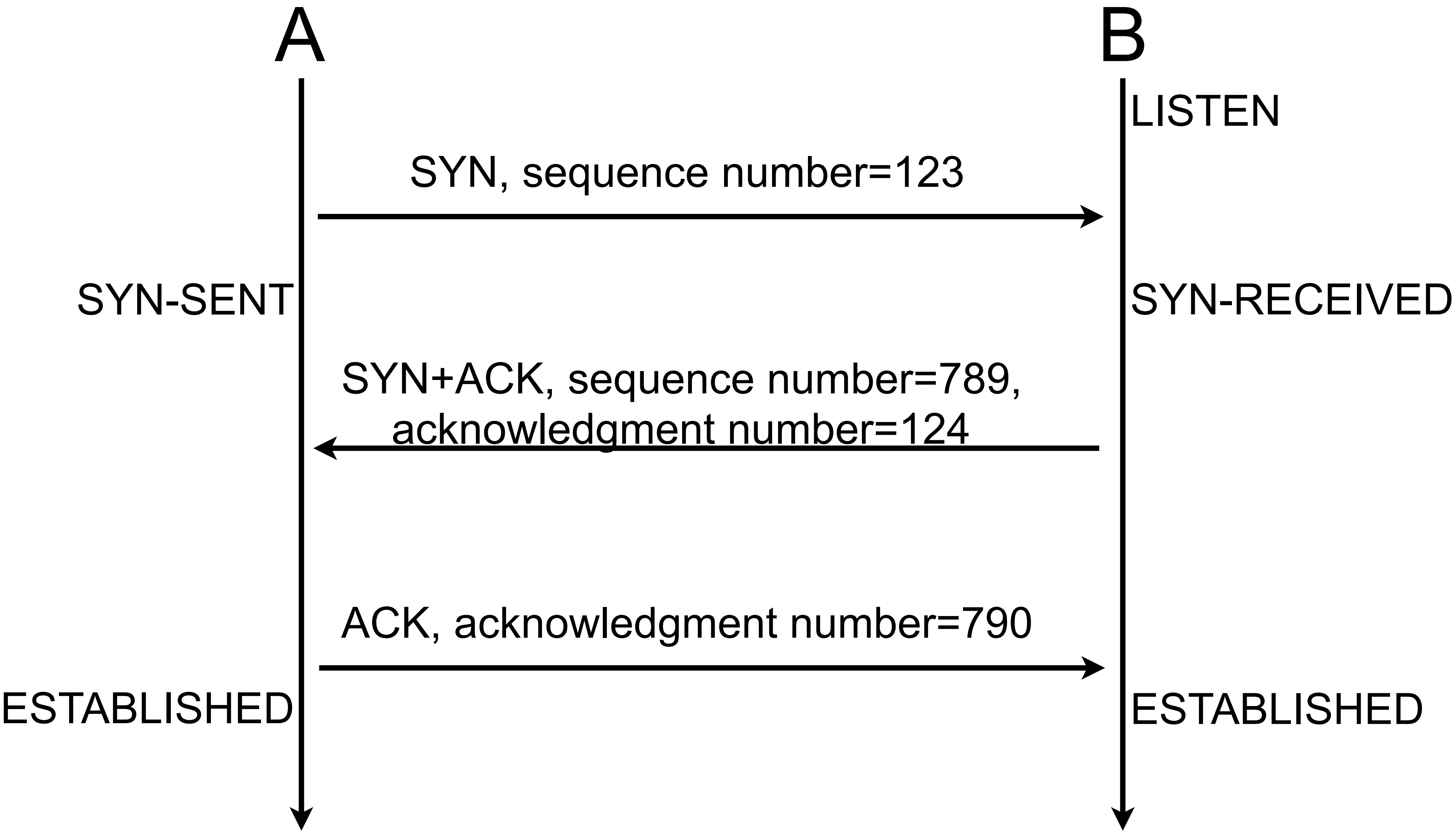
SYN-SENT

SYN-RECEIVED

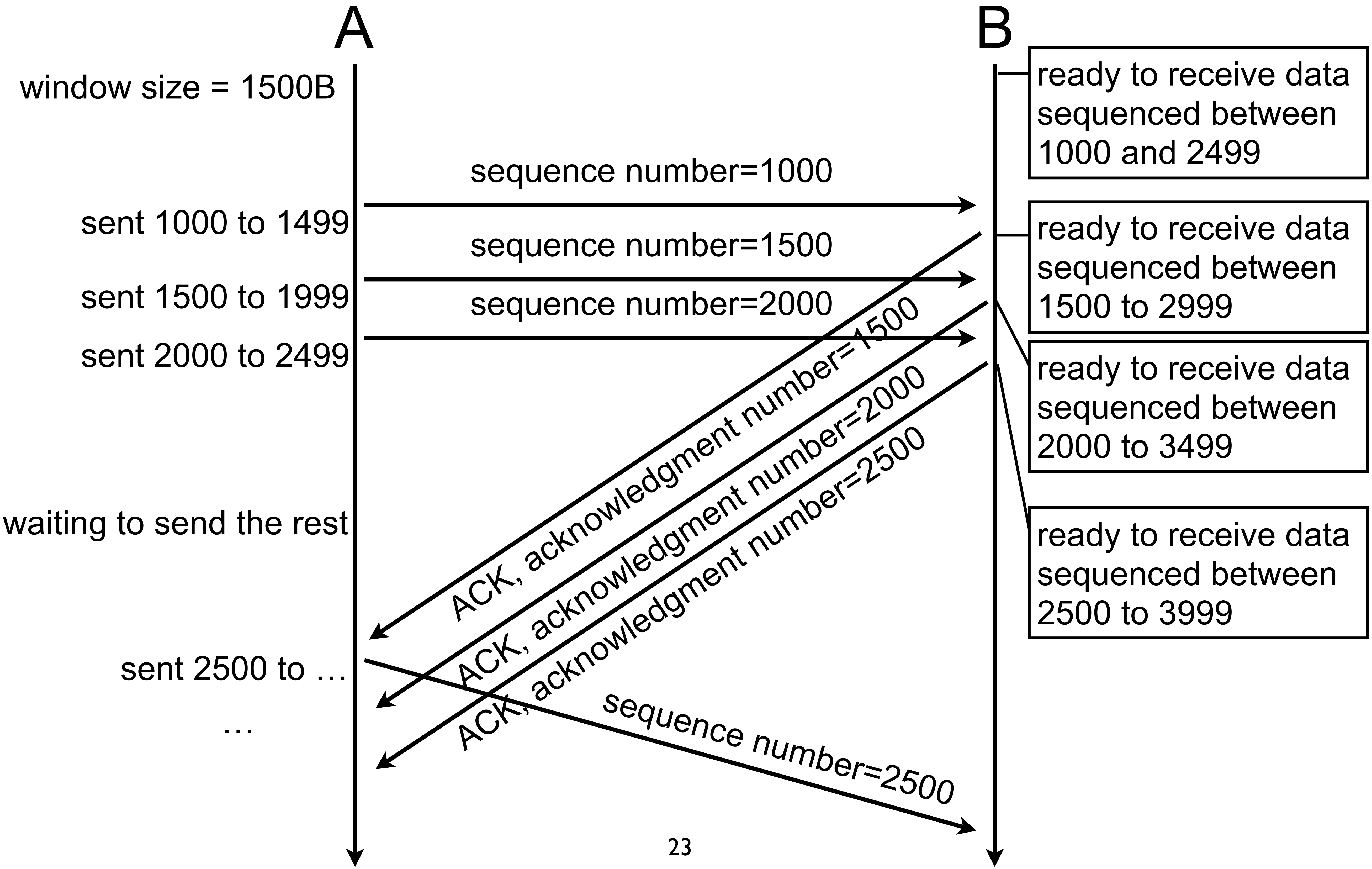
SYN+ACK, sequence number=789, acknowledgment number=124

ACK, acknowledgment number=790

TCP connection establishment


A
B
LISTEN
SYN, sequence number=123
SYN-SENT
SYN-RECEIVED
SYN+ACK, sequence number=789,
acknowledgment number=124
ACK, acknowledgment number=790
ESTABLISHED
ESTABLISHED

TCP data transfer



A

B

window size = 1500B

ready to receive data sequenced between 1000 and 2499

sequence number=1000

sent 1000 to 1499

ready to receive data sequenced between 1500 to 2999

sequence number=1500

sent 1500 to 1999

sequence number=2000

sent 2000 to 2499

ready to receive data sequenced between 2000 to 3499

ACK, acknowledgment number=1500

ACK, acknowledgment number=2000

ACK, acknowledgment number=2500

waiting to send the rest

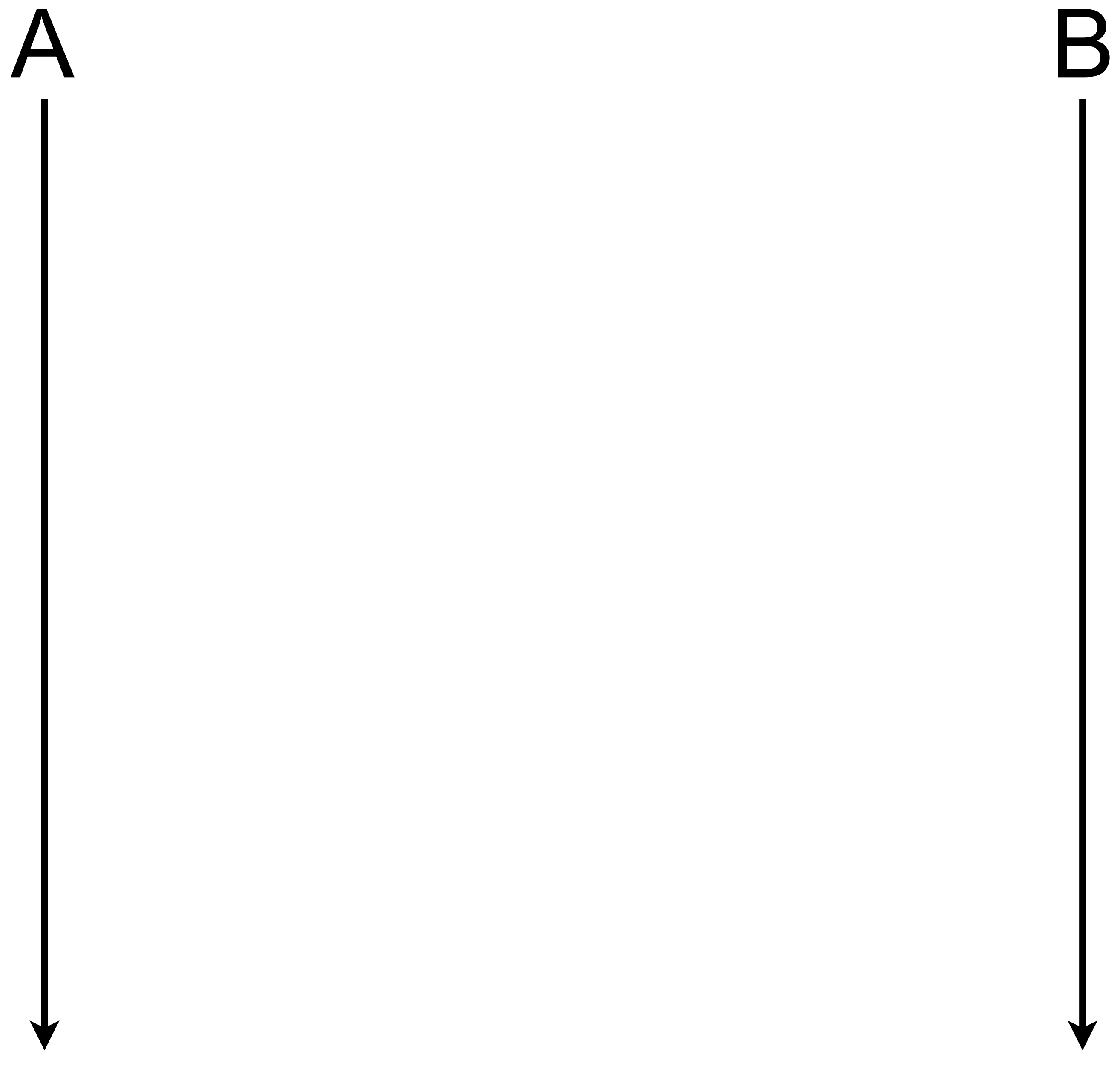
ready to receive data sequenced between 2500 to 3999

sent 2500 to …

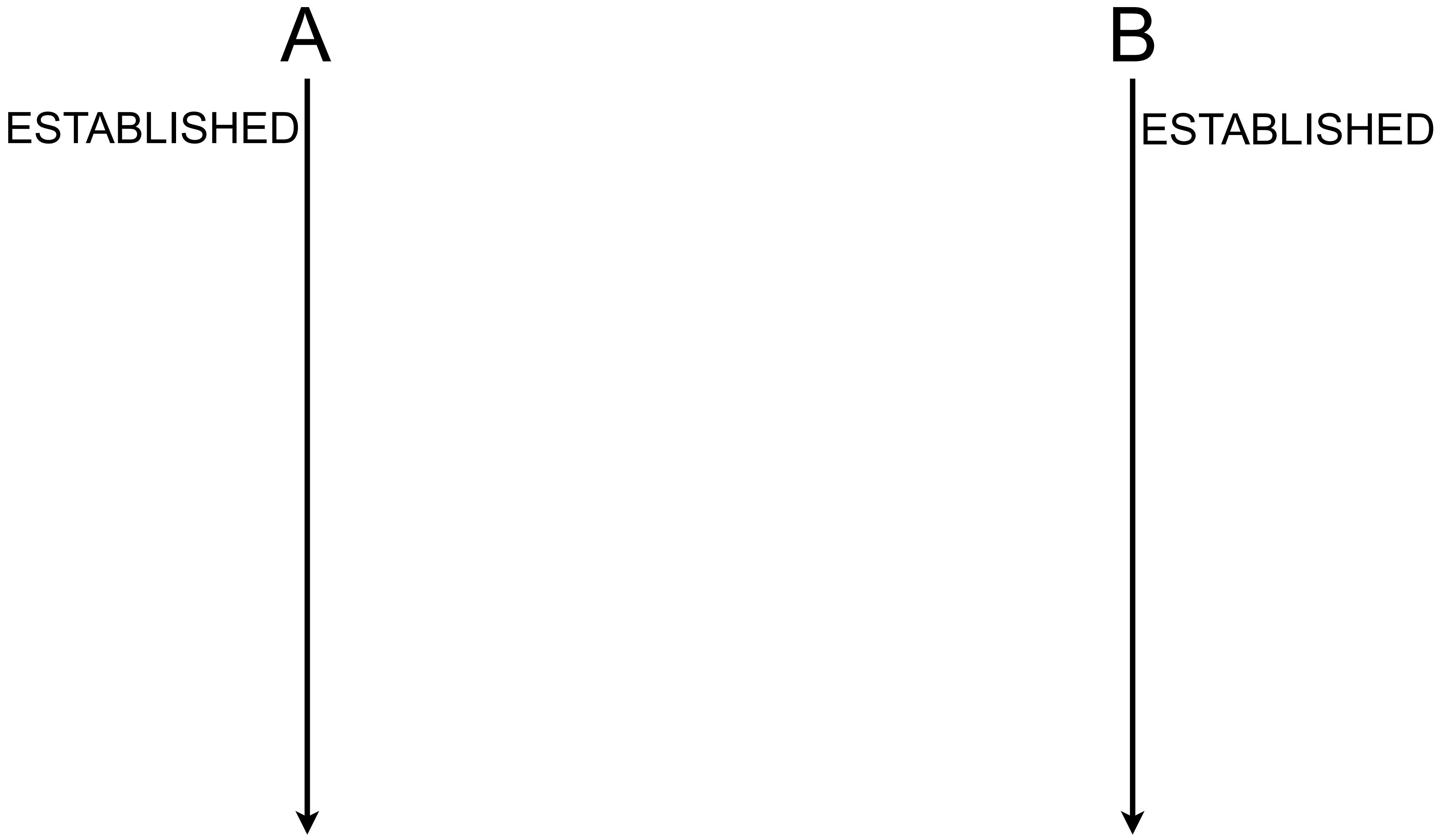
…

sequence number=2500

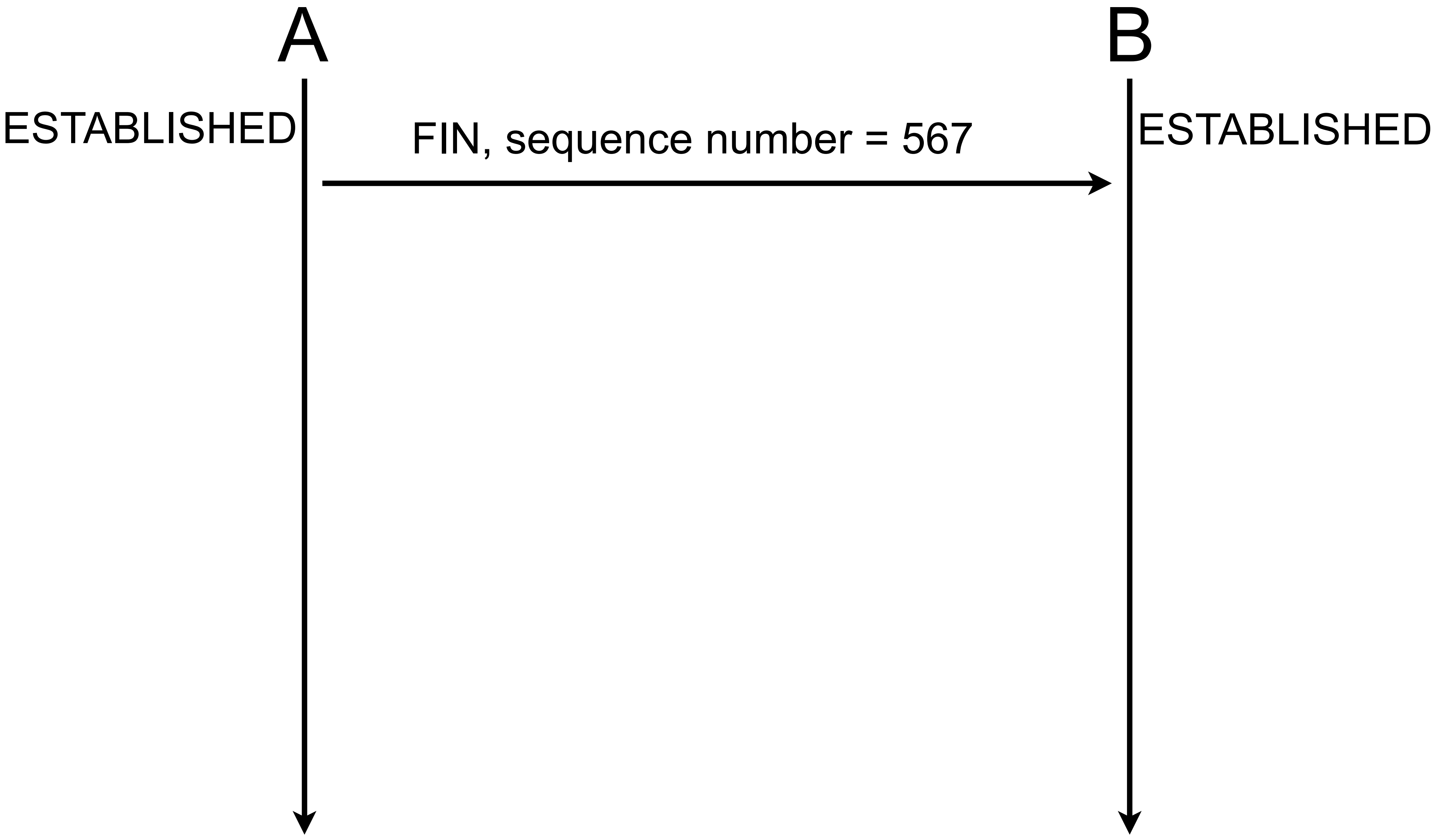
TCP connection termination



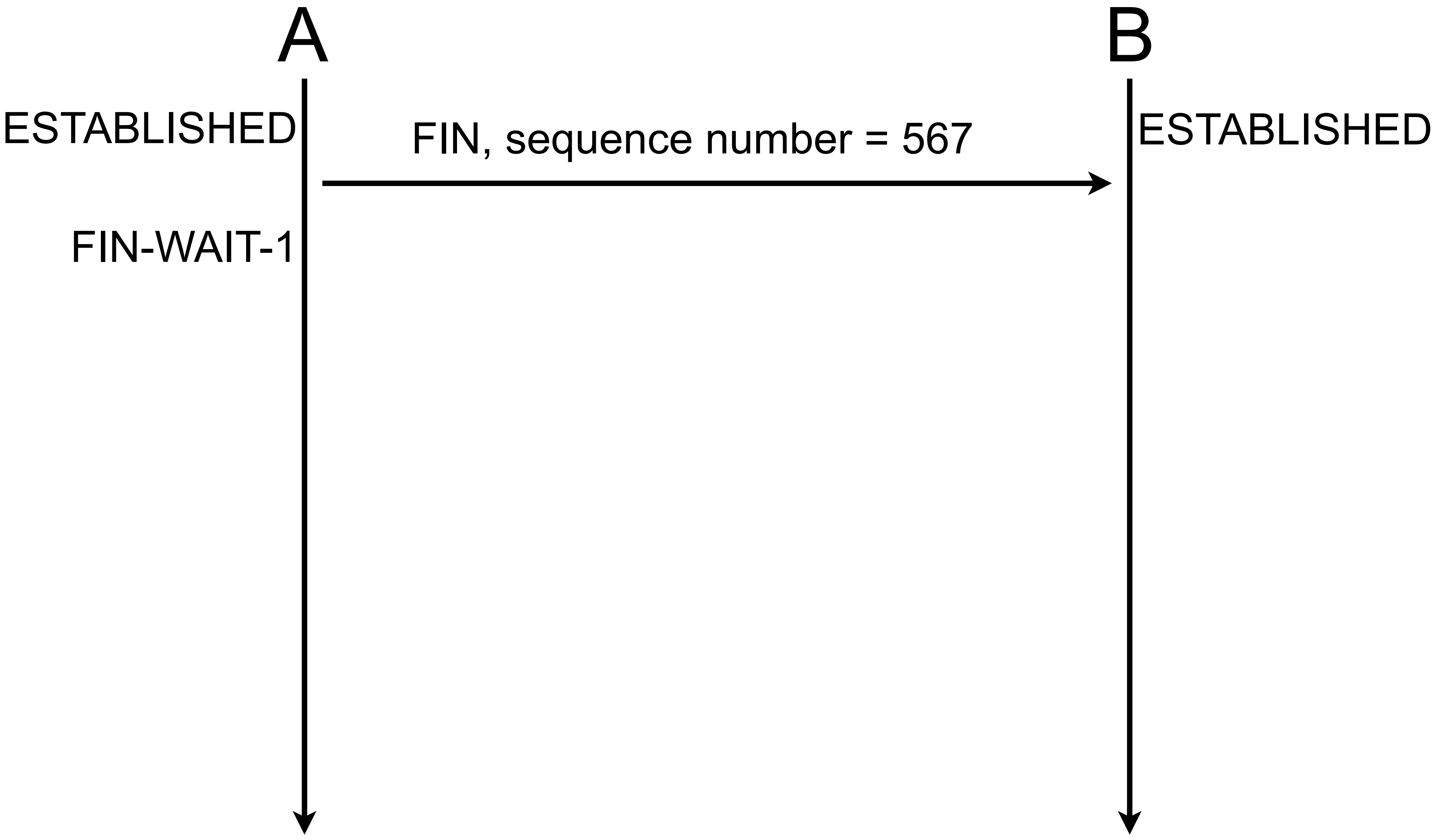
TCP connection termination



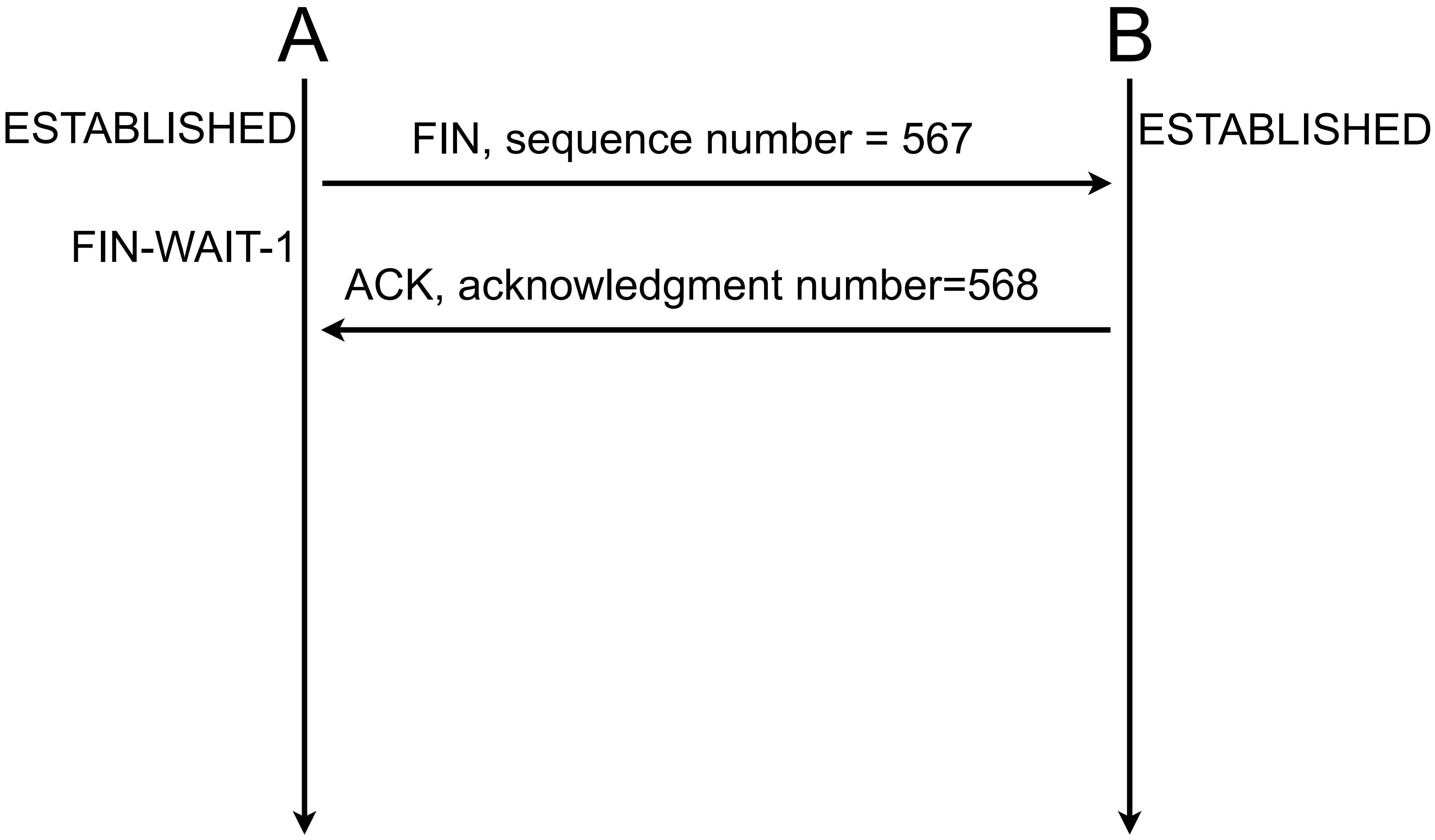
TCP connection termination



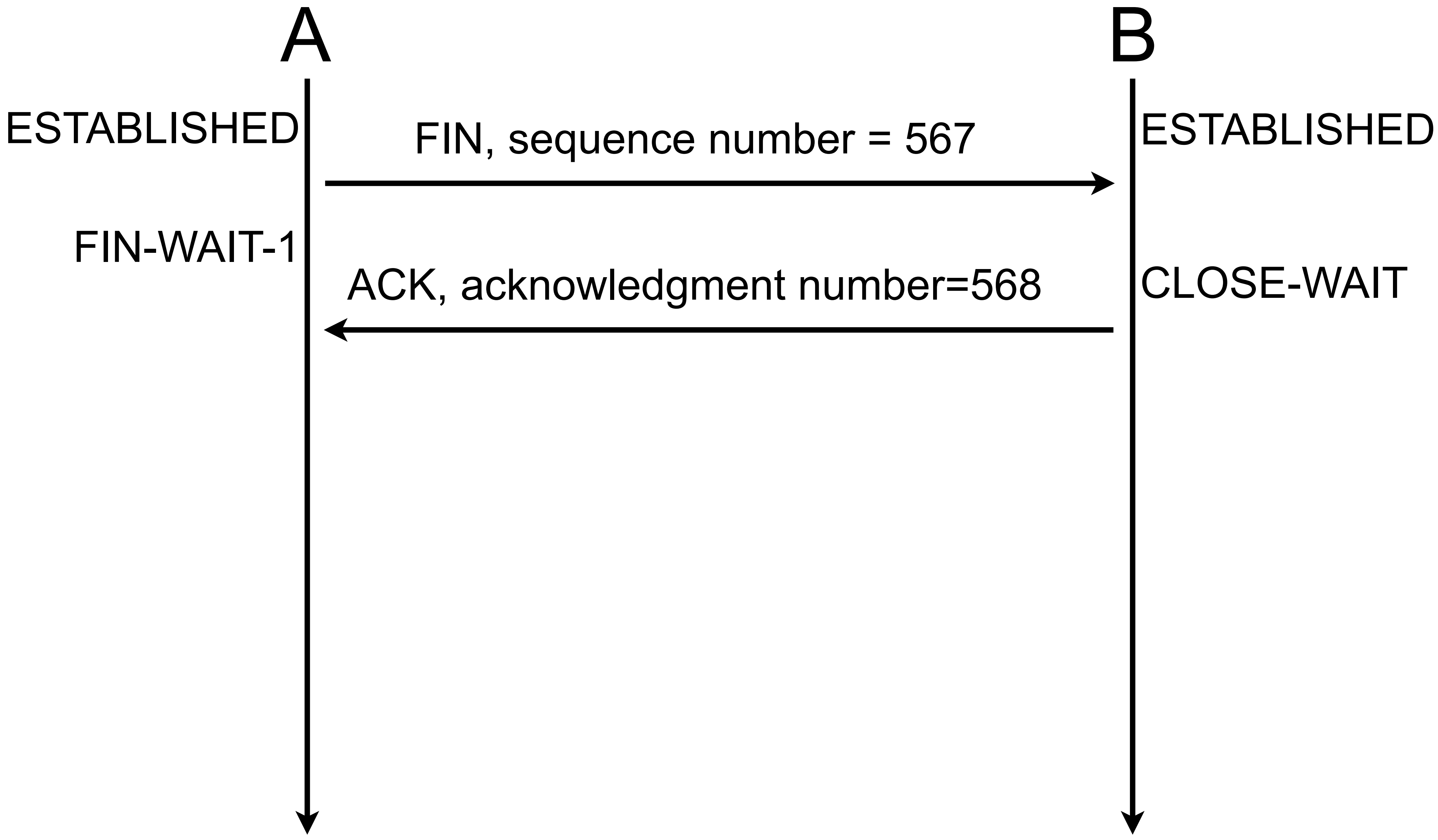
TCP connection termination



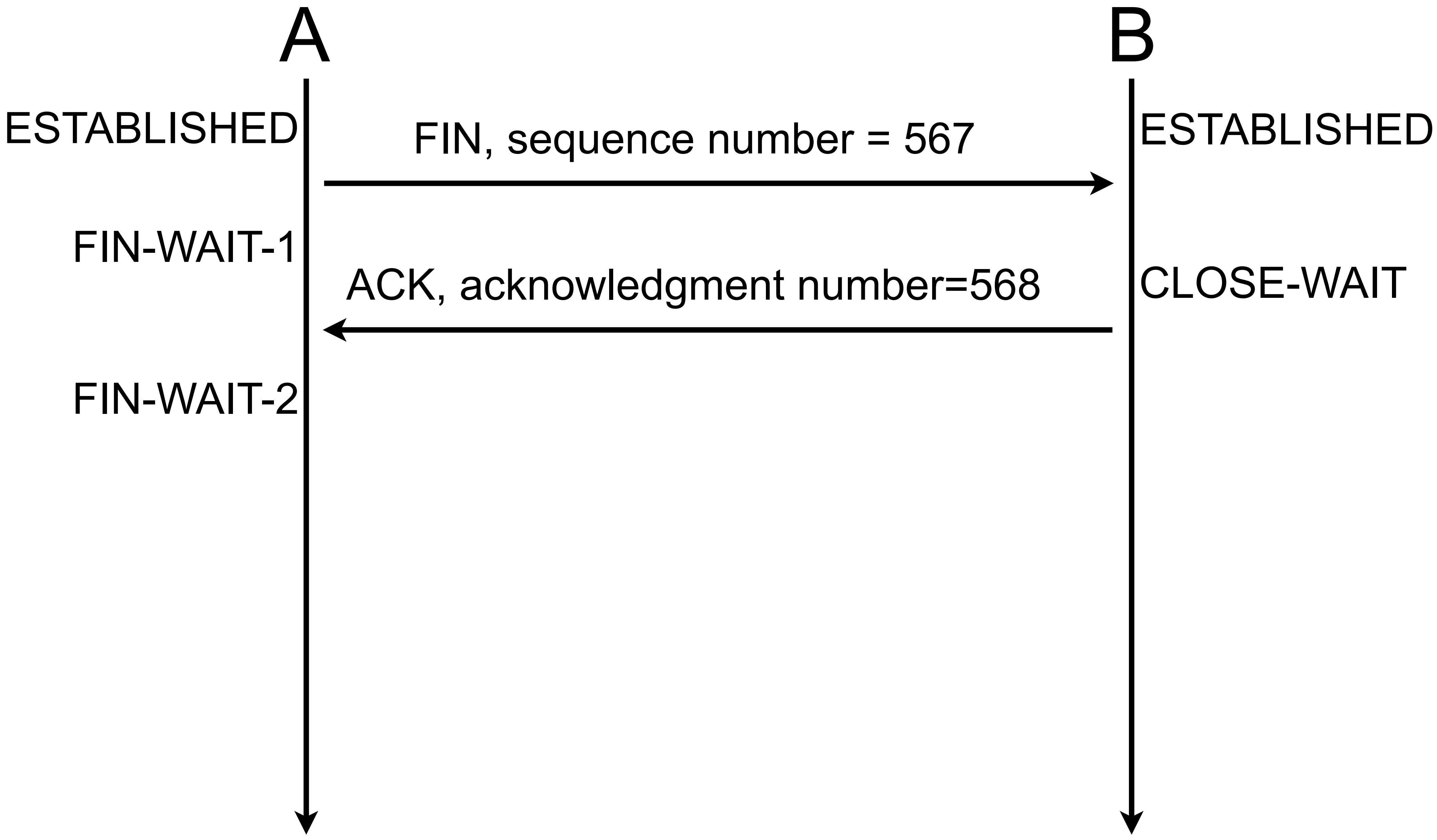
TCP connection termination



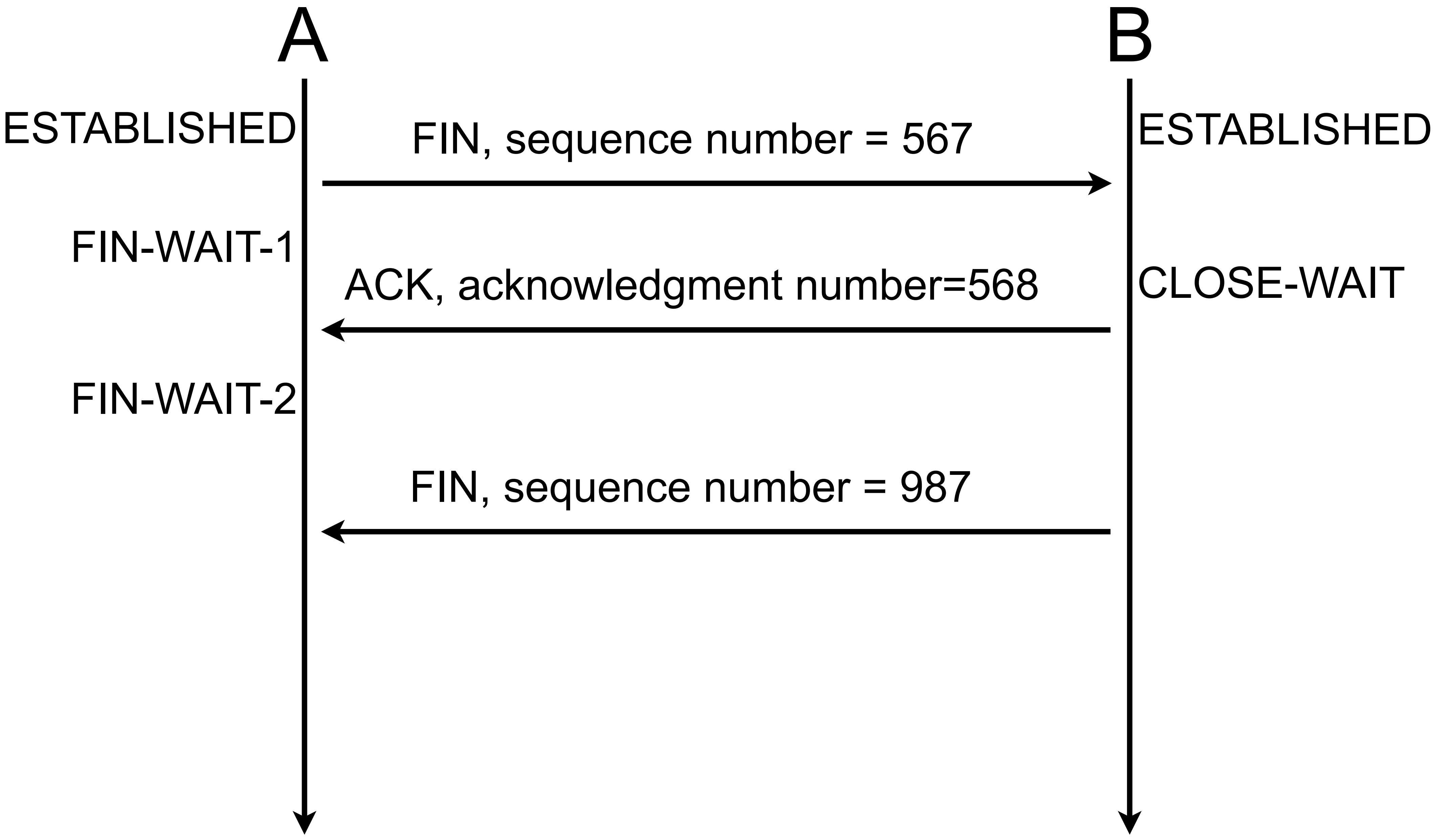
TCP connection termination



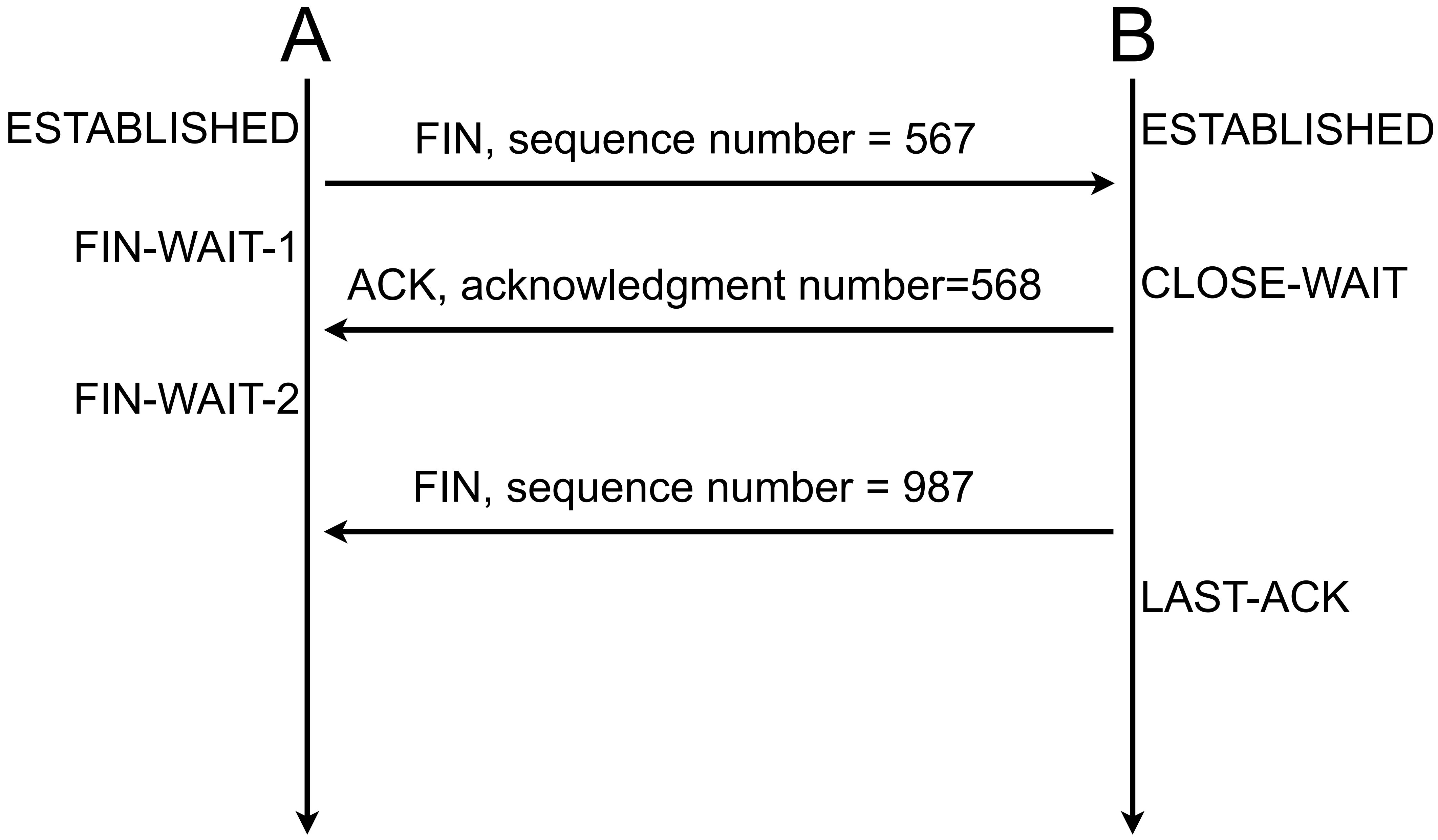
TCP connection termination



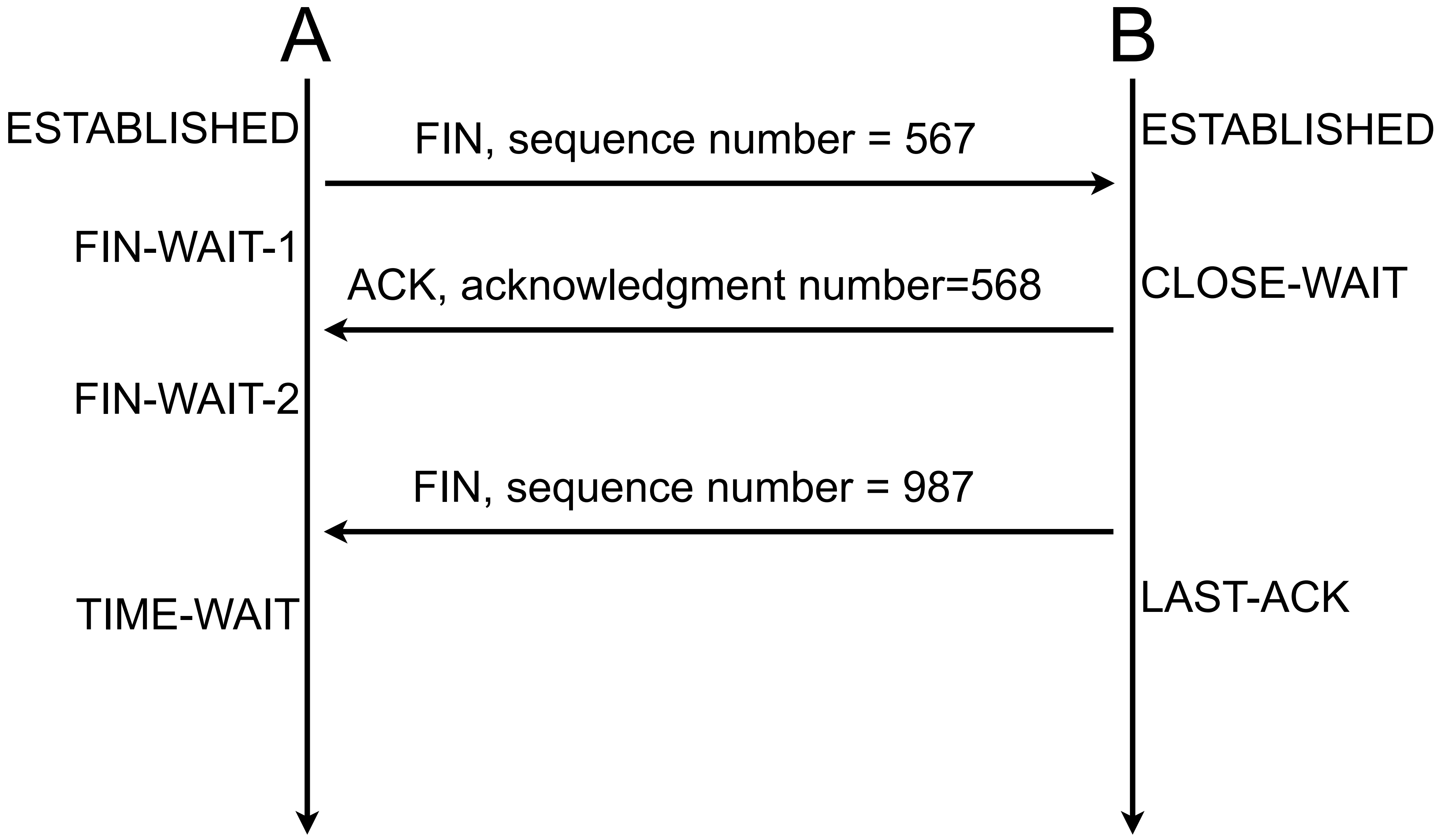
TCP connection termination


A
B
ESTABLISHED
FIN, sequence number = 567
ESTABLISHED
FIN-WAIT-1
ACK, acknowledgment number=568
CLOSE-WAIT
FIN-WAIT-2
FIN, sequence number = 987

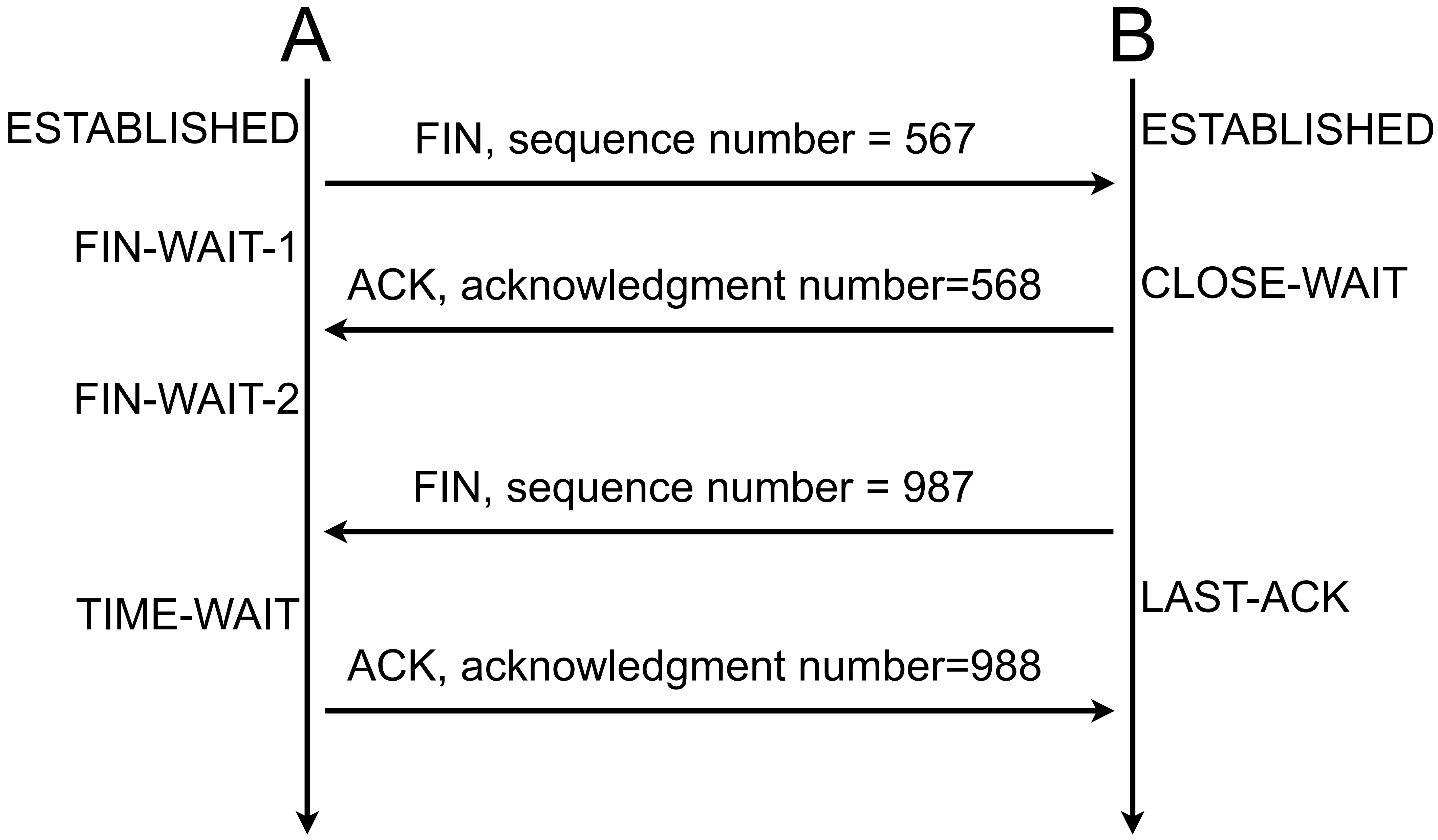
TCP connection termination


A
B
ESTABLISHED
FIN, sequence number = 567
ESTABLISHED
FIN-WAIT-1
ACK, acknowledgment number=568
CLOSE-WAIT
FIN-WAIT-2
FIN, sequence number = 987
LAST-ACK

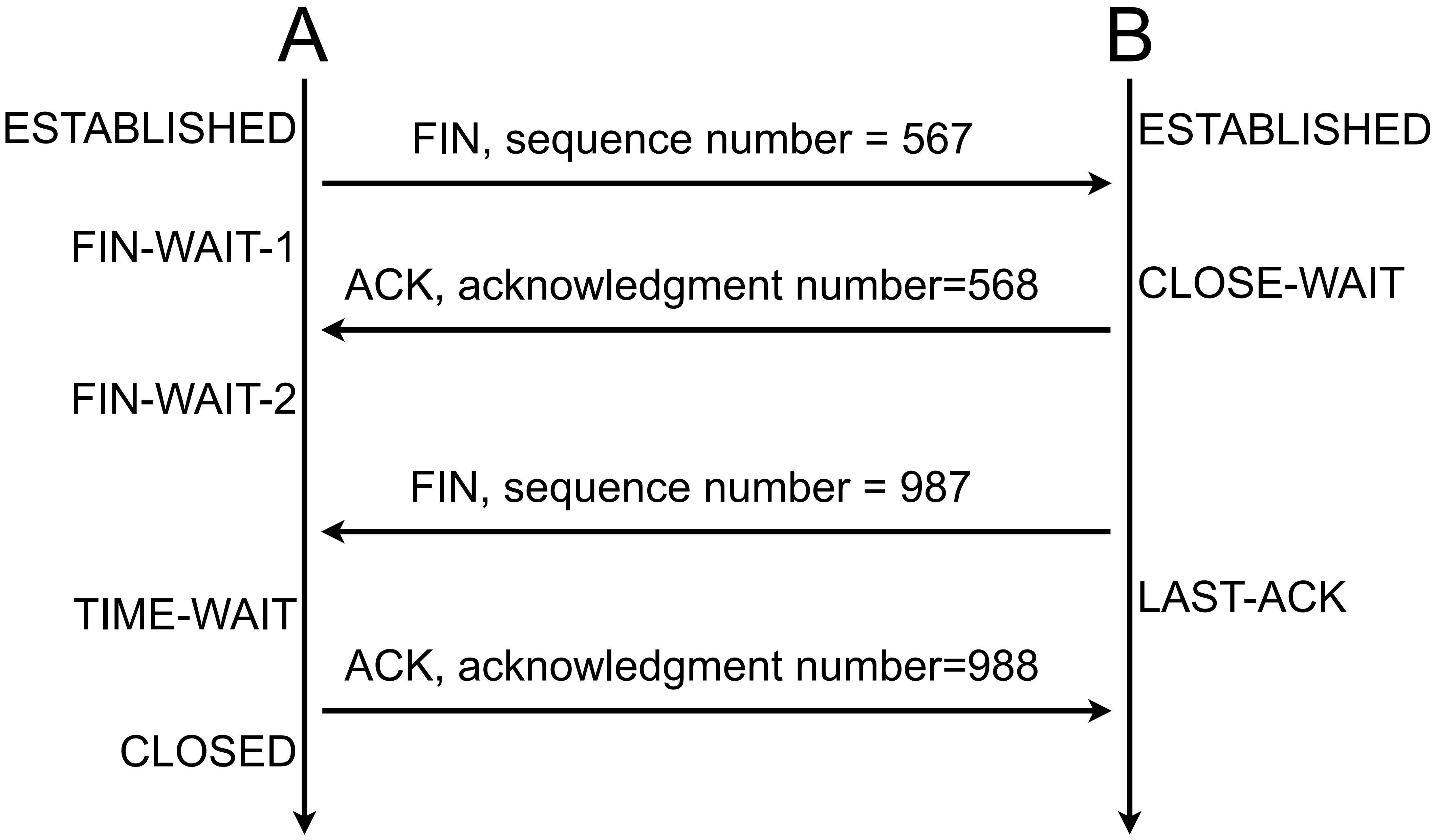
TCP connection termination


A
B
ESTABLISHED
FIN, sequence number = 567
ESTABLISHED
FIN-WAIT-1
ACK, acknowledgment number=568
CLOSE-WAIT
FIN-WAIT-2
FIN, sequence number = 987
LAST-ACK
TIME-WAIT

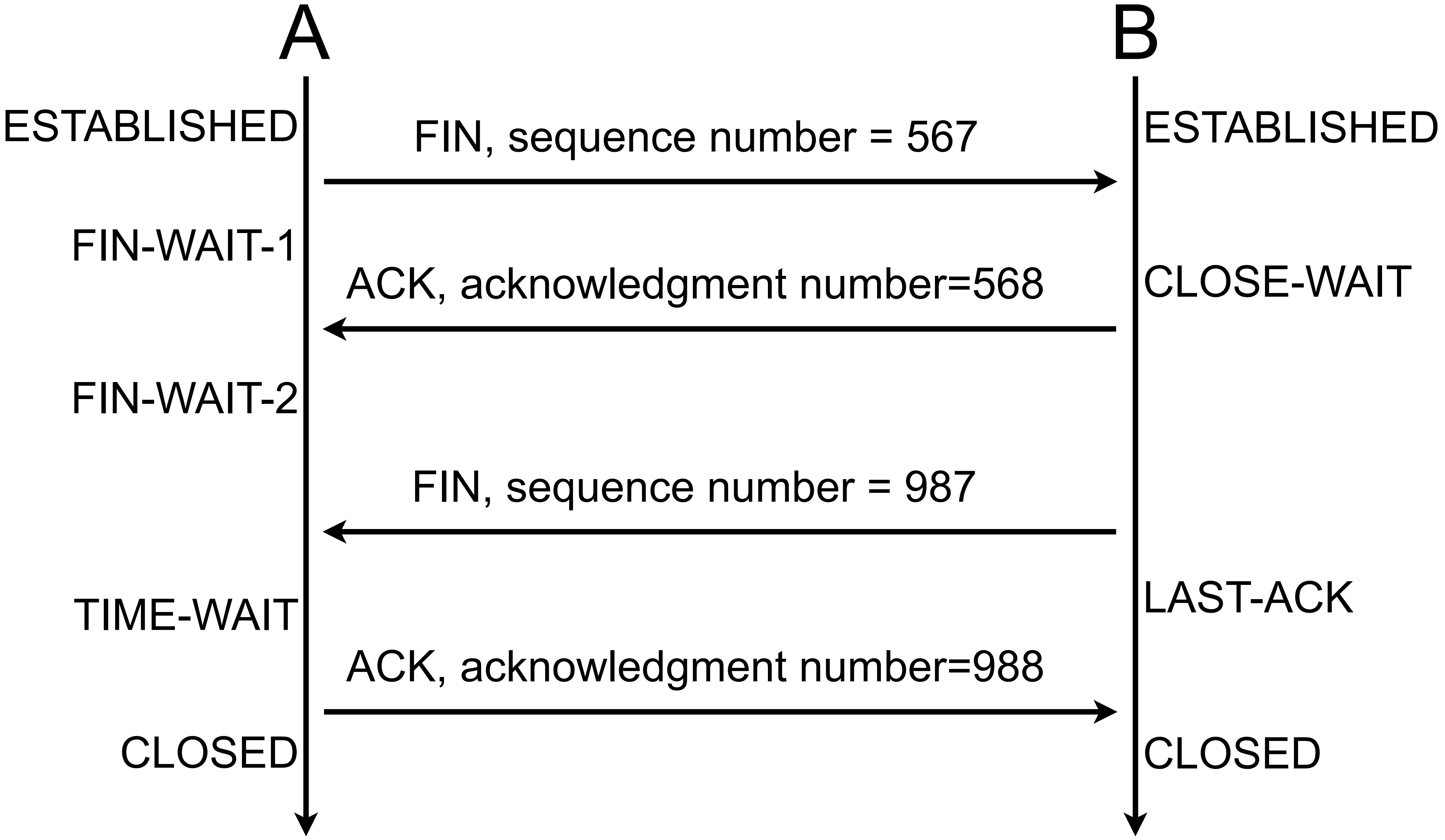
TCP connection termination


A
B
ESTABLISHED
FIN, sequence number = 567
ESTABLISHED
FIN-WAIT-1
ACK, acknowledgment number=568
CLOSE-WAIT
FIN-WAIT-2
FIN, sequence number = 987
LAST-ACK
TIME-WAIT
ACK, acknowledgment number=988

TCP connection termination


A
B
ESTABLISHED
FIN, sequence number = 567
ESTABLISHED
FIN-WAIT-1
ACK, acknowledgment number=568
CLOSE-WAIT
FIN-WAIT-2
FIN, sequence number = 987
TIME-WAIT
LAST-ACK
ACK, acknowledgment number=988
CLOSED

TCP connection termination



A

B

ESTABLISHED

FIN, sequence number = 567

ESTABLISHED

FIN-WAIT-1

ACK, acknowledgment number=568

CLOSE-WAIT

FIN-WAIT-2

FIN, sequence number = 987

LAST-ACK

TIME-WAIT

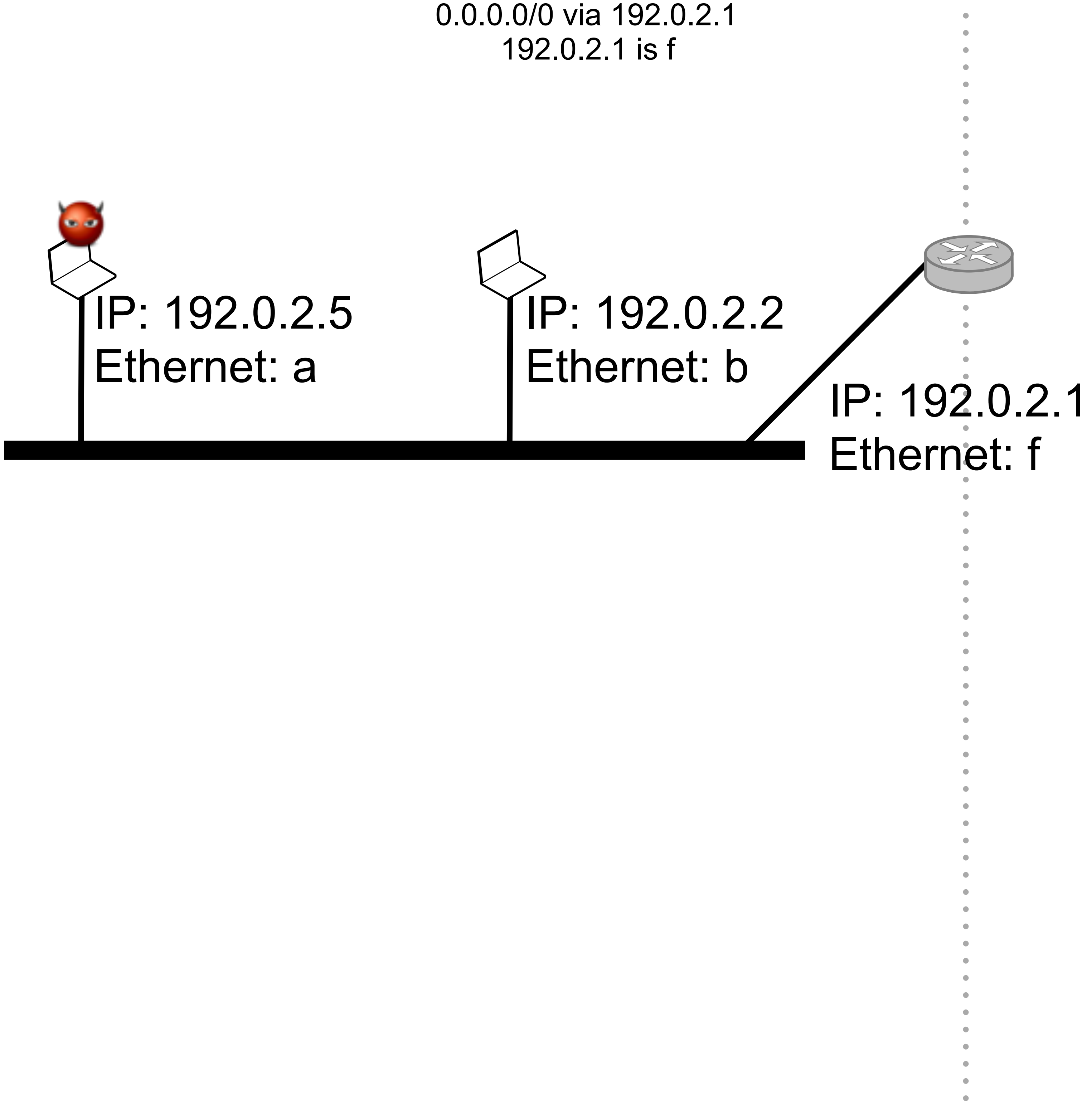
ACK, acknowledgment number=988

CLOSED

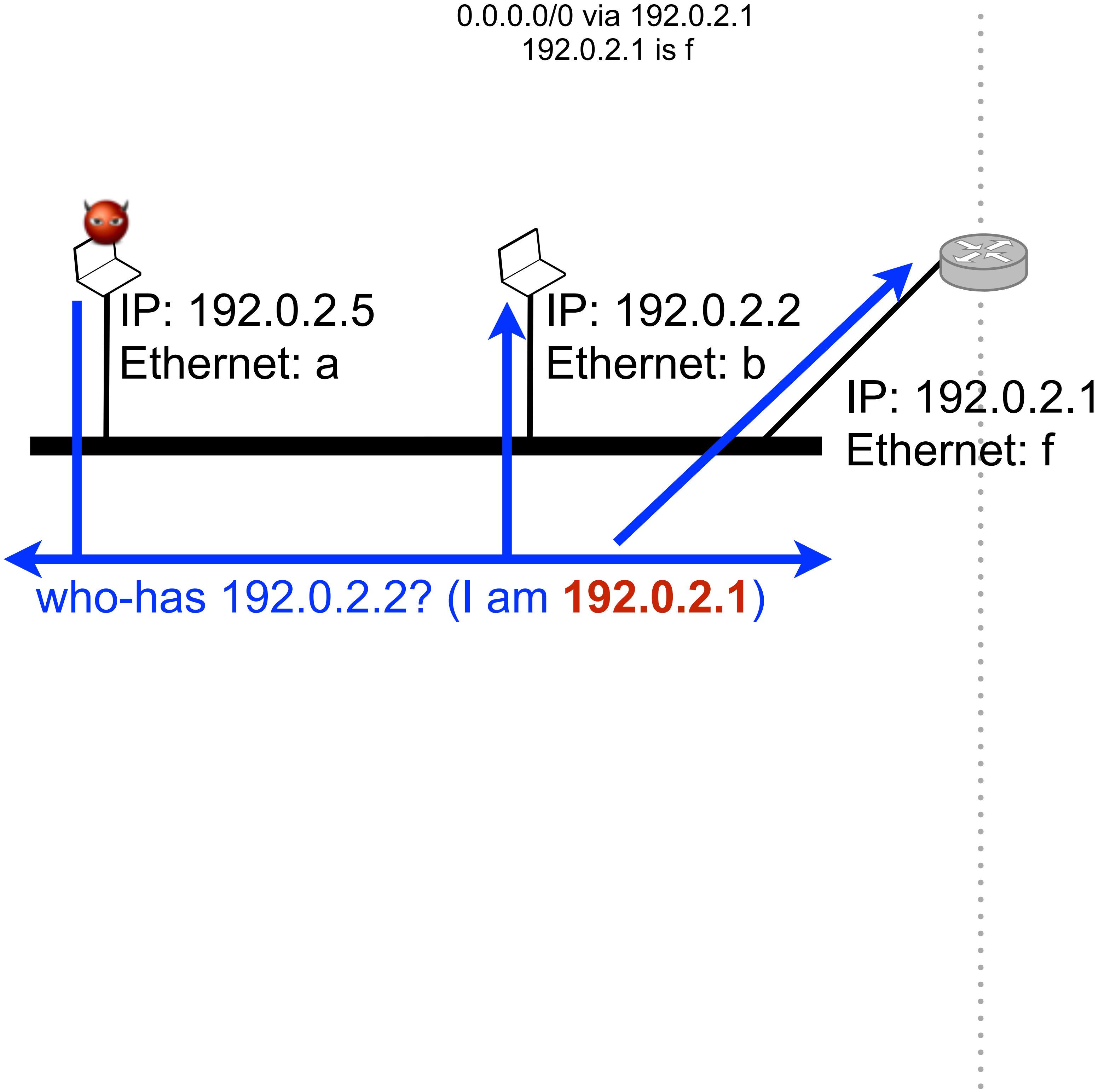
CLOSED

Threats by the example

ARP poisoning



ARP poisoning



0.0.0.0/0 via 192.0.2.1
192.0.2.1 is f

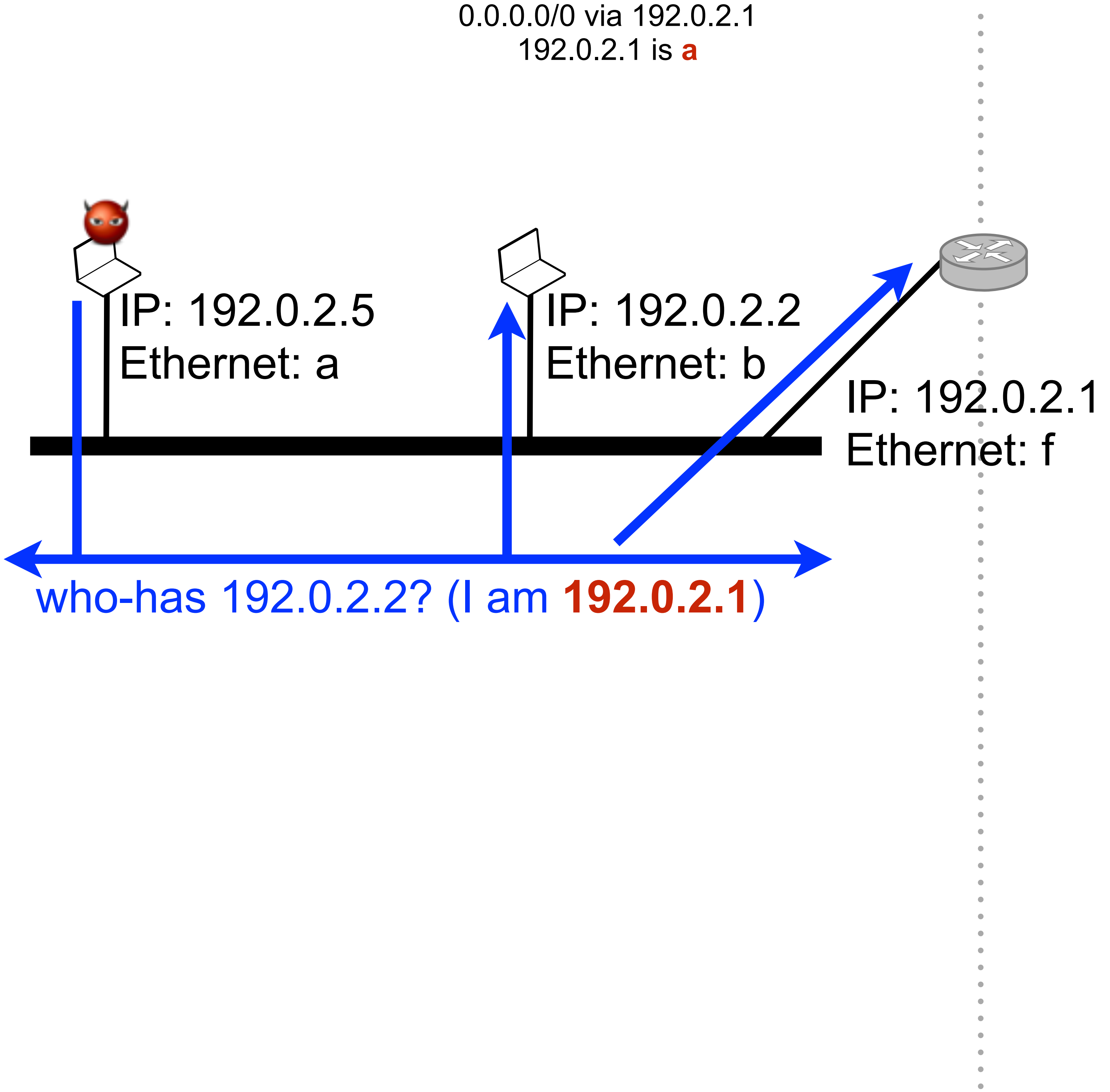
IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

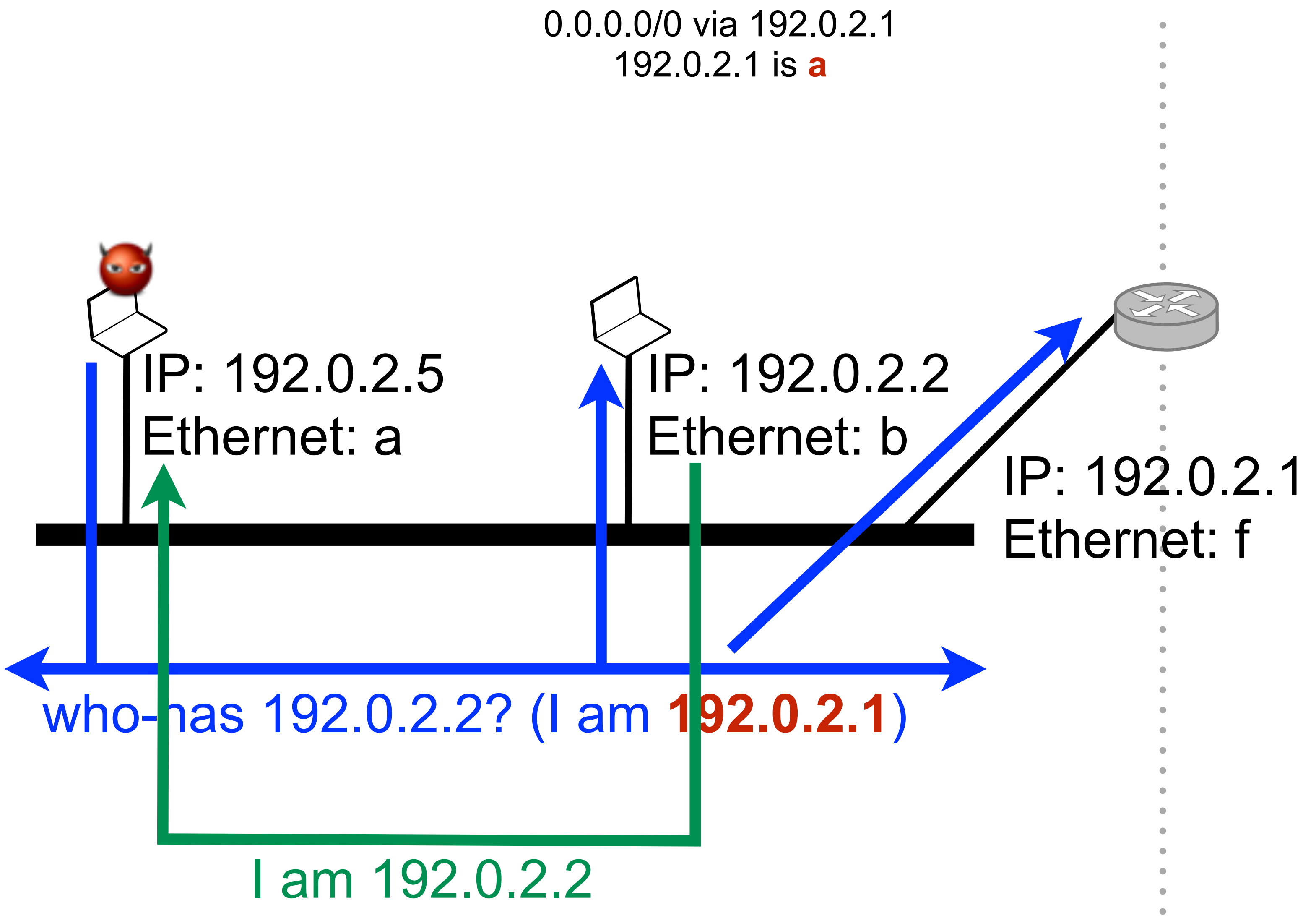
IP: 192.0.2.1
Ethernet: f

who-has 192.0.2.2? (I am **192.0.2.1**)

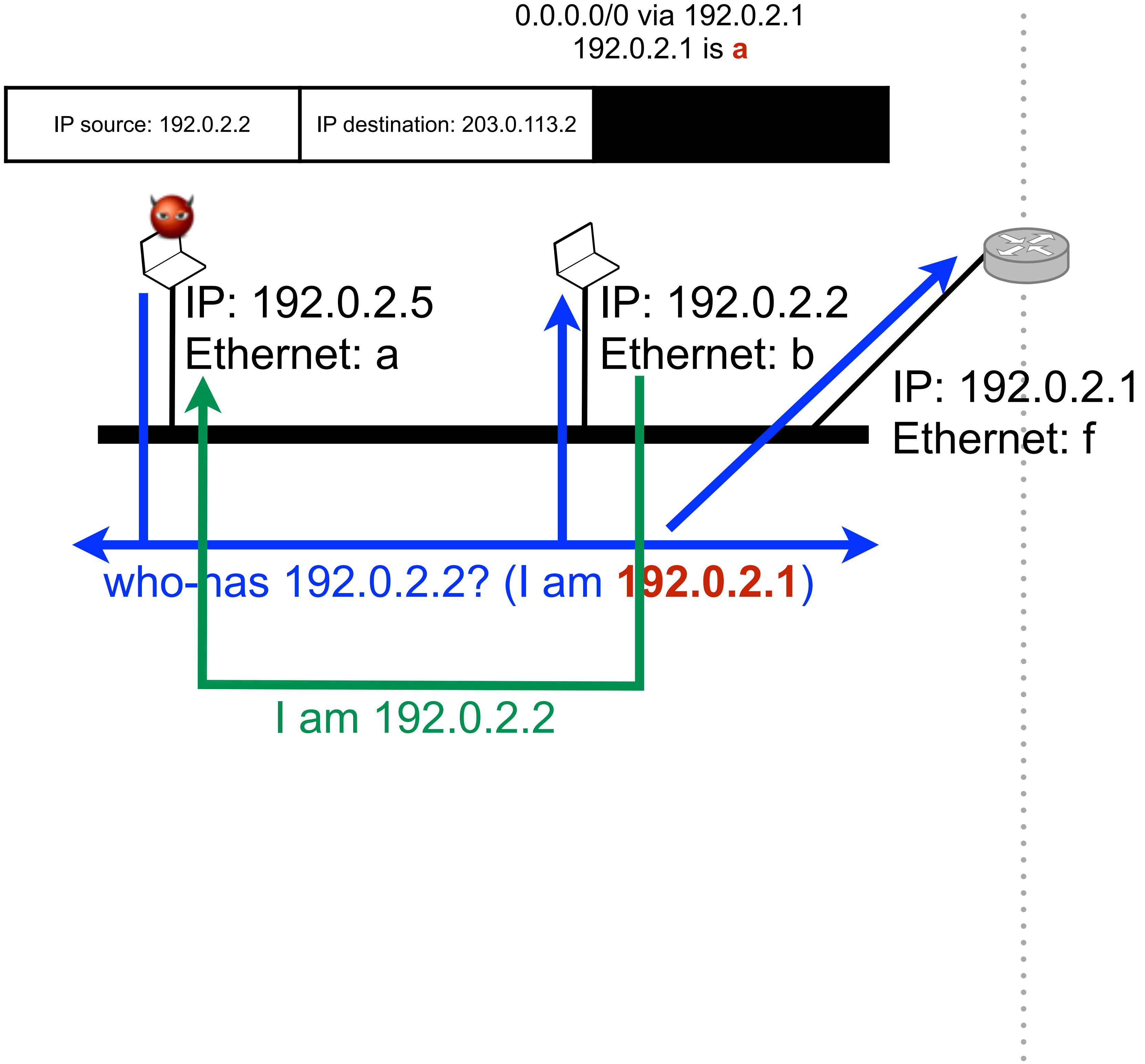
ARP poisoning



ARP poisoning



ARP poisoning



0.0.0.0/0 via 192.0.2.1
192.0.2.1 is **a**

IP source: 192.0.2.2

IP destination: 203.0.113.2

IP: 192.0.2.5
Ethernet: a

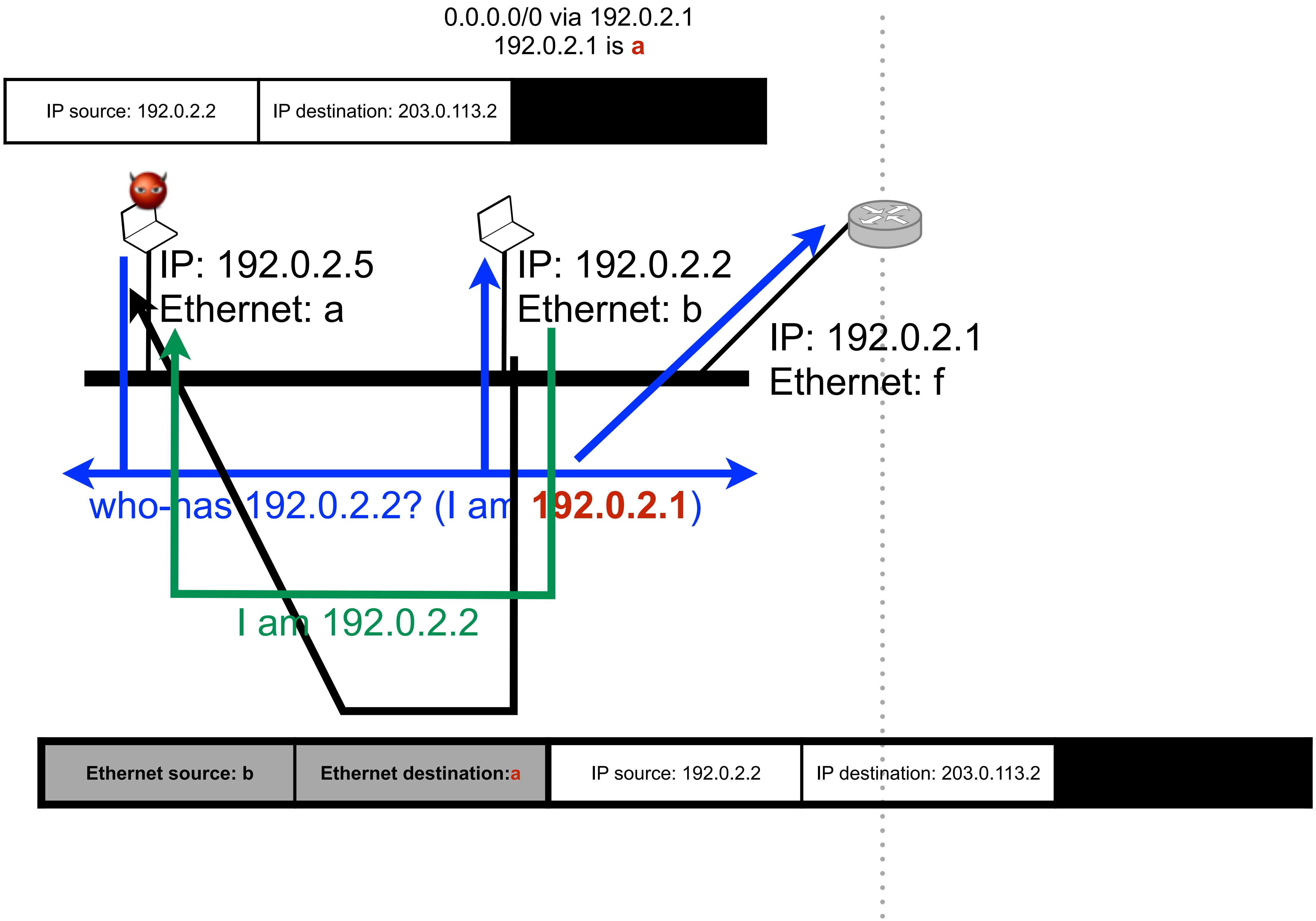
IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.1
Ethernet: f

who-has 192.0.2.2? (I am **192.0.2.1**)

I am 192.0.2.2

ARP poisoning



0.0.0.0/0 via 192.0.2.1
192.0.2.1 is **a**

IP source: 192.0.2.2	IP destination: 203.0.113.2

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.1
Ethernet: f

who-has 192.0.2.2? (I am **192.0.2.1**)

I am 192.0.2.2

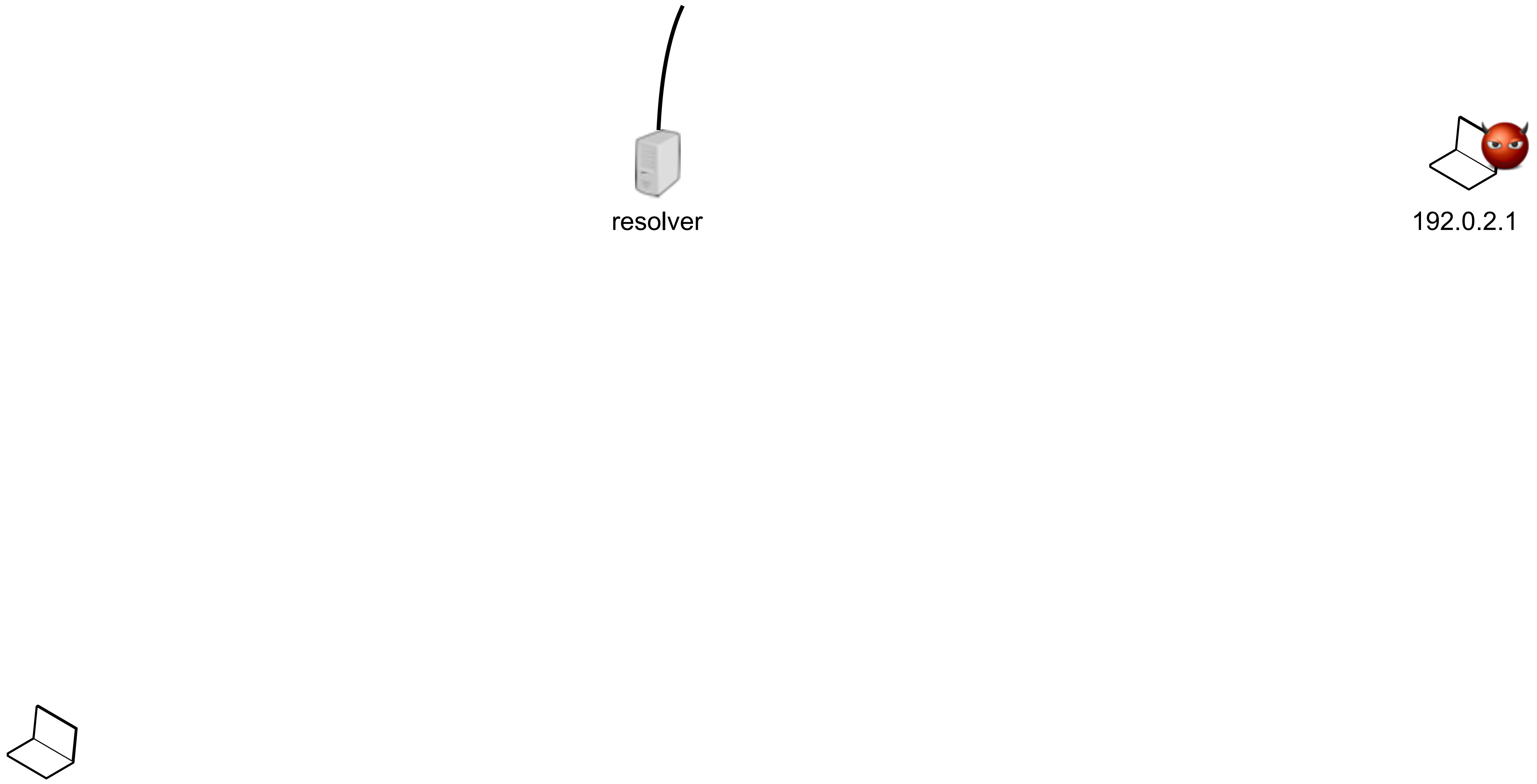
Ethernet source: b	**Ethernet destination:a**	IP source: 192.0.2.2	IP destination: 203.0.113.2

Why does it work?

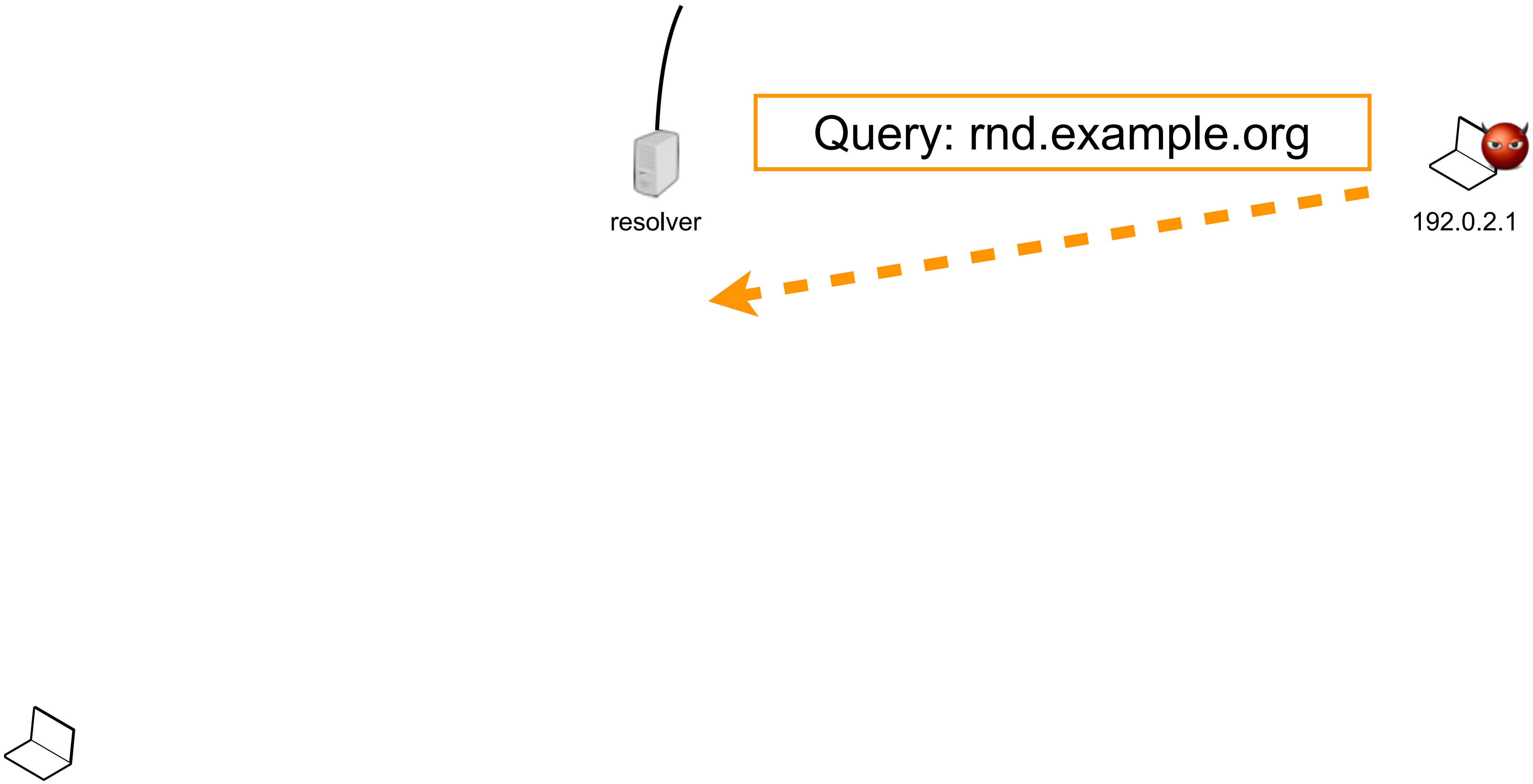
Why does it work?

- Conceptual vulnerability
  - using non-requested information as ground truth is dangerous
  - using non-authenticated information is dangerous

DNS cache poisoning

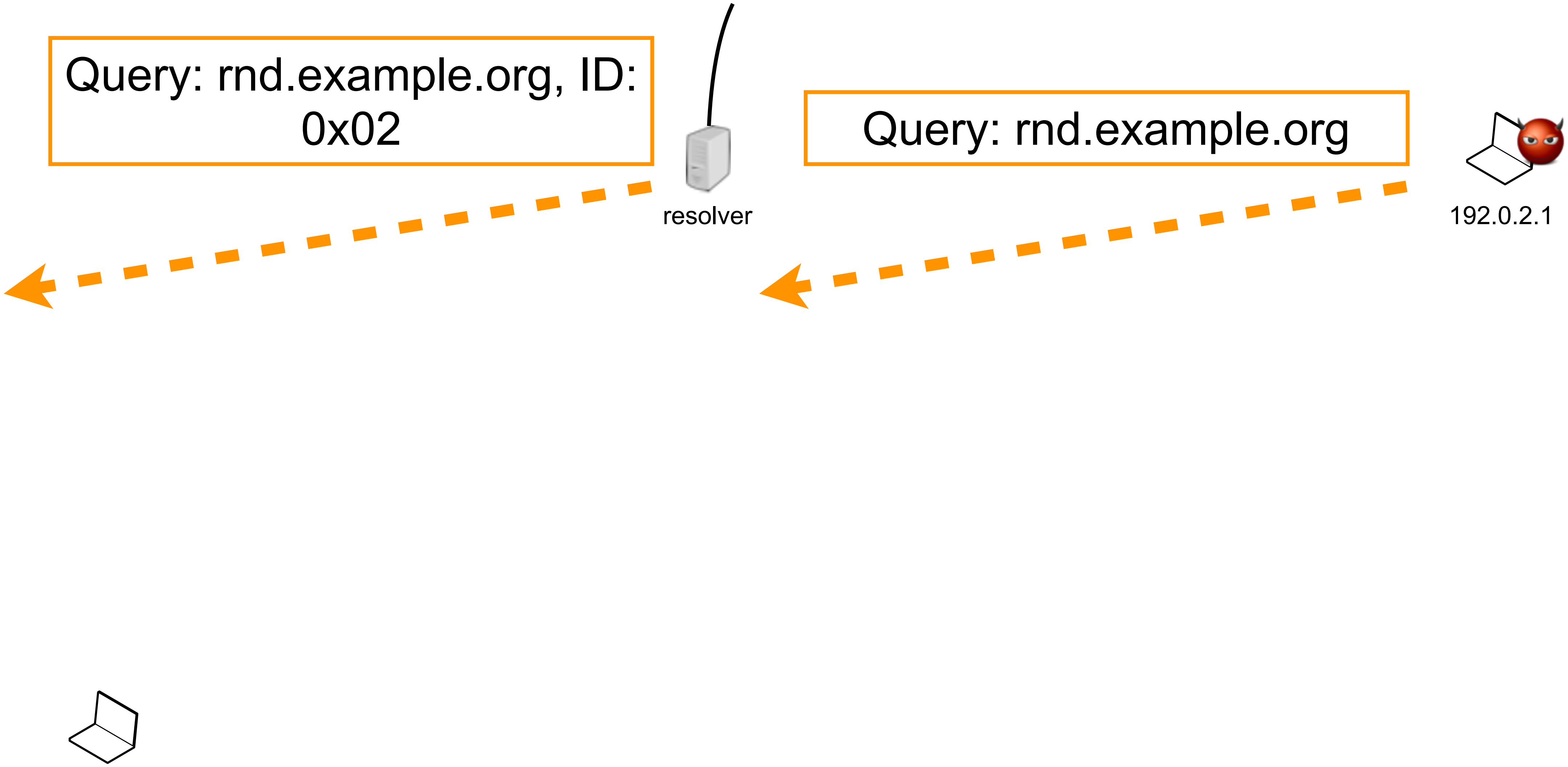


DNS cache poisoning



Query: rnd.example.org

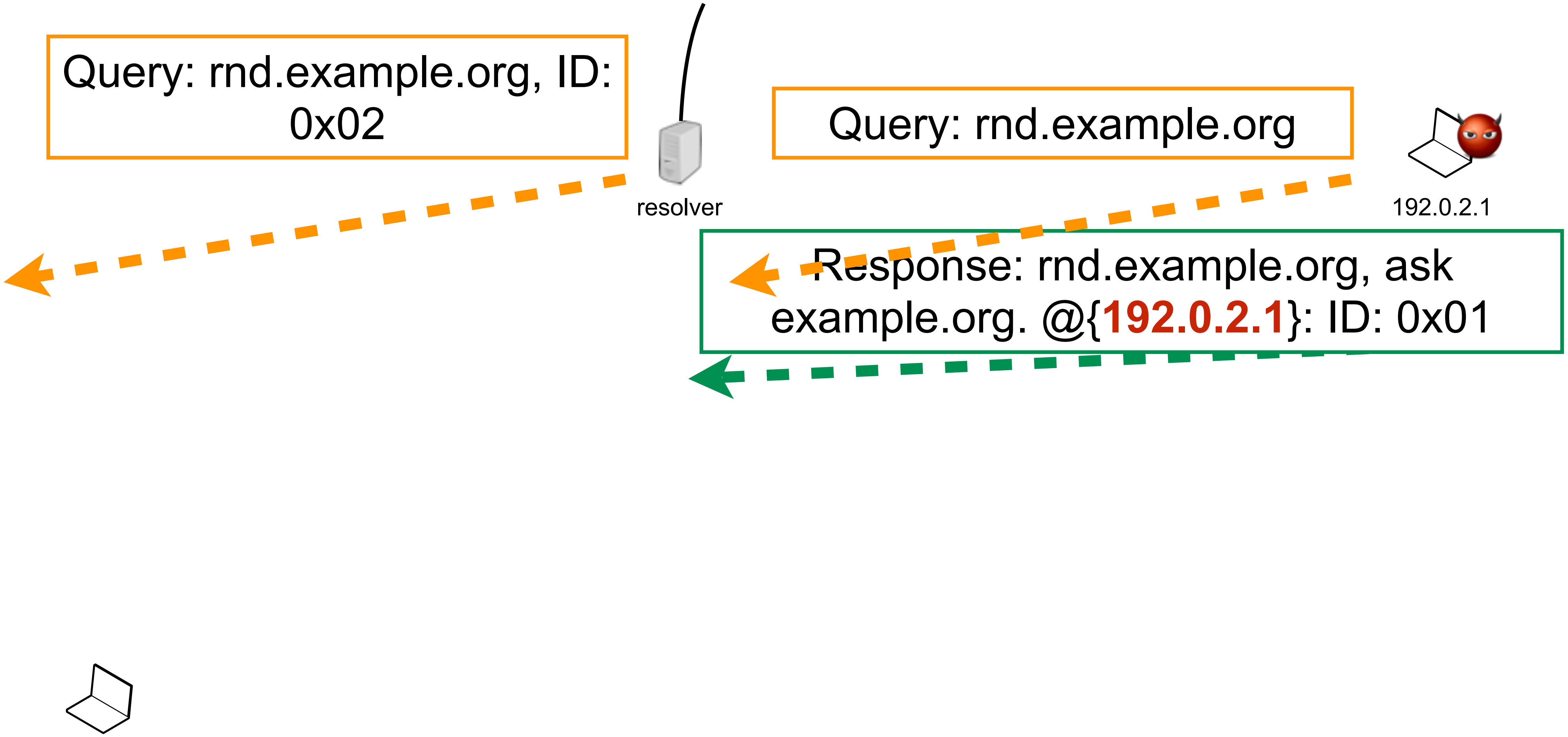
DNS cache poisoning



Query: rnd.example.org, ID: 0x02

Query: rnd.example.org

DNS cache poisoning

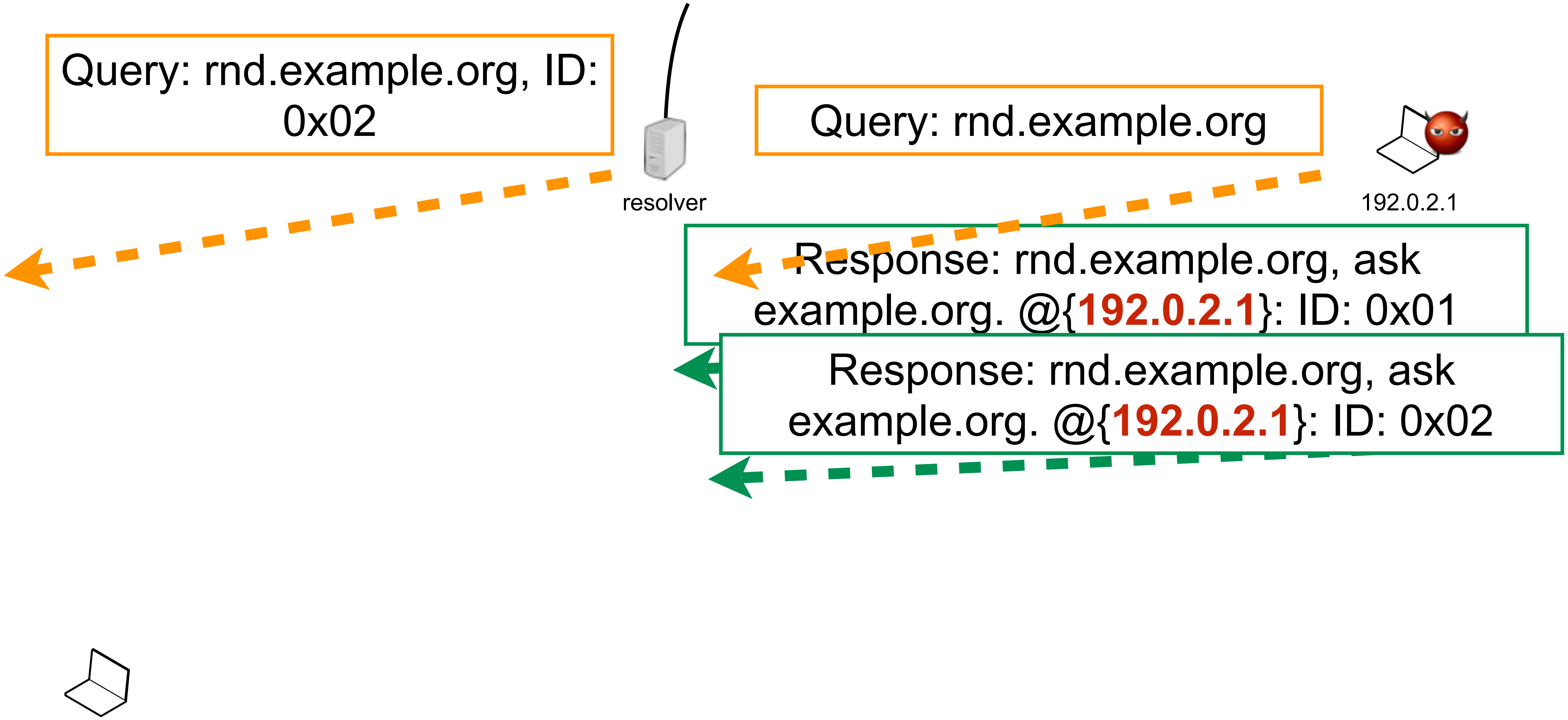


Query: rnd.example.org, ID: 0x02

Query: rnd.example.org

Response: rnd.example.org, ask example.org. @{**192.0.2.1**}: ID: 0x01

DNS cache poisoning



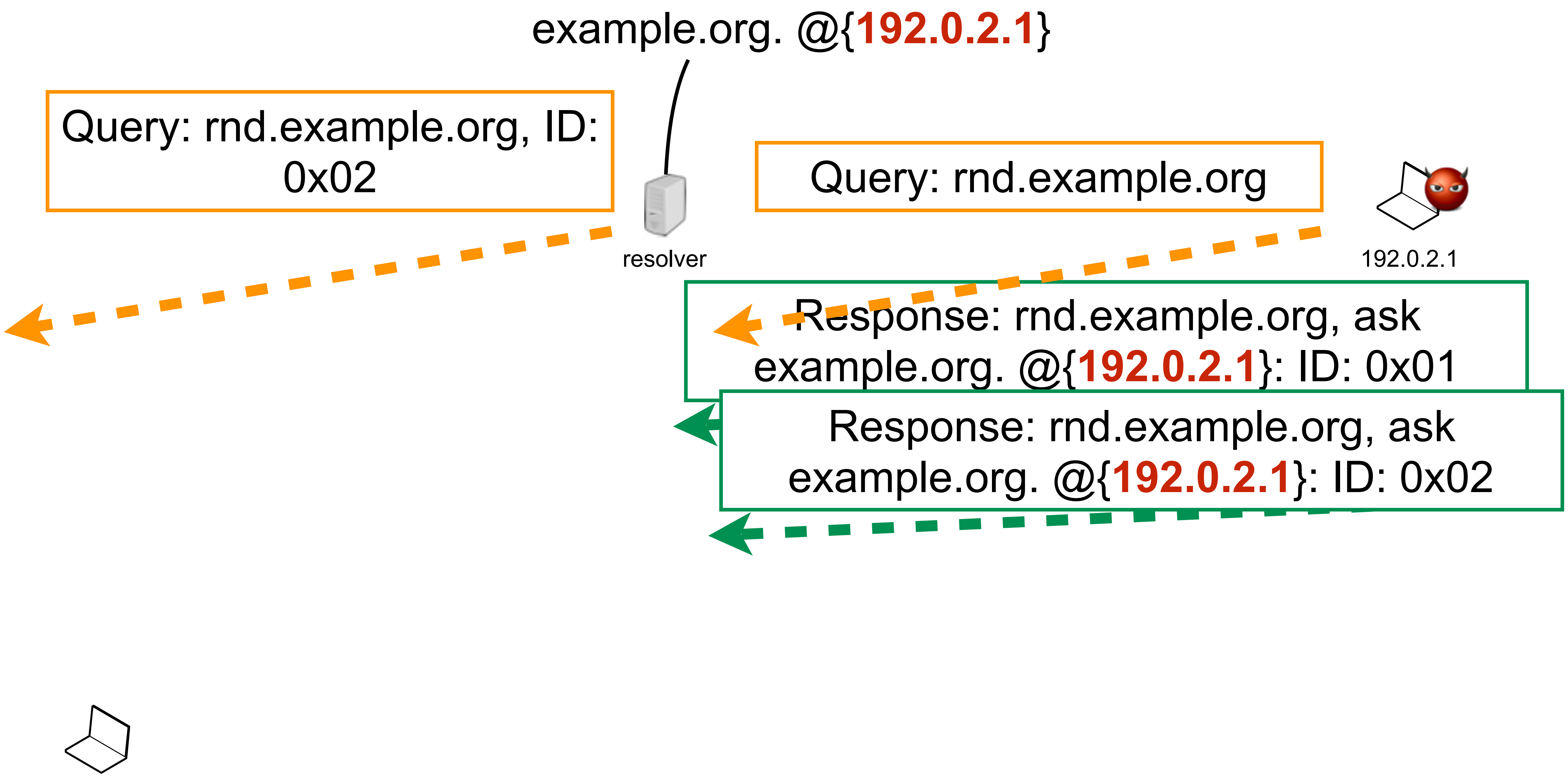
Query: rnd.example.org, ID: 0x02

Query: rnd.example.org

Response: rnd.example.org, ask example.org. @{**192.0.2.1**}: ID: 0x01

Response: rnd.example.org, ask example.org. @{**192.0.2.1**}: ID: 0x02

DNS cache poisoning



example.org. @{192.0.2.1}

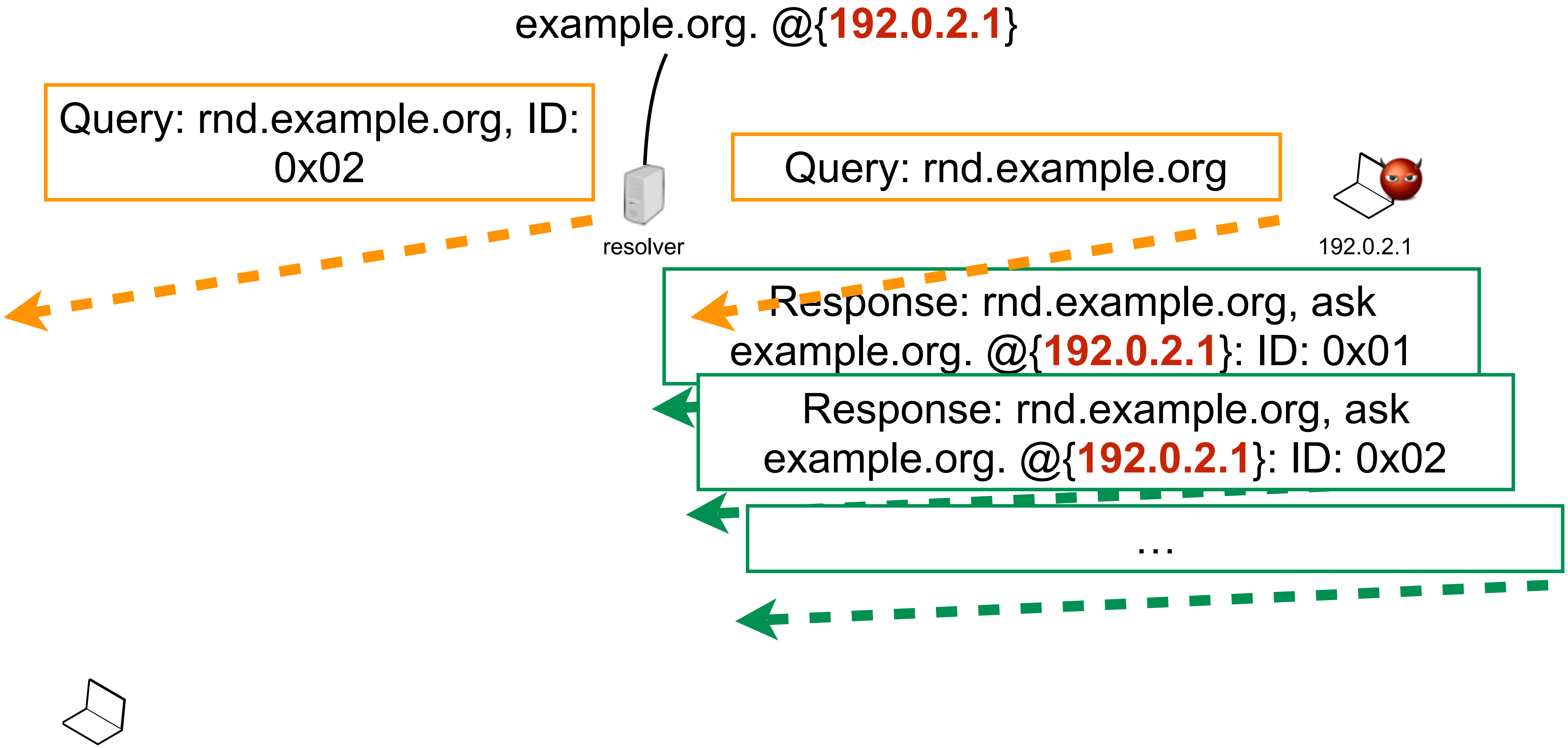
Query: rnd.example.org, ID: 0x02

Query: rnd.example.org

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x01

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02

DNS cache poisoning



example.org. @{192.0.2.1}

Query: rnd.example.org, ID: 0x02

resolver

Query: rnd.example.org

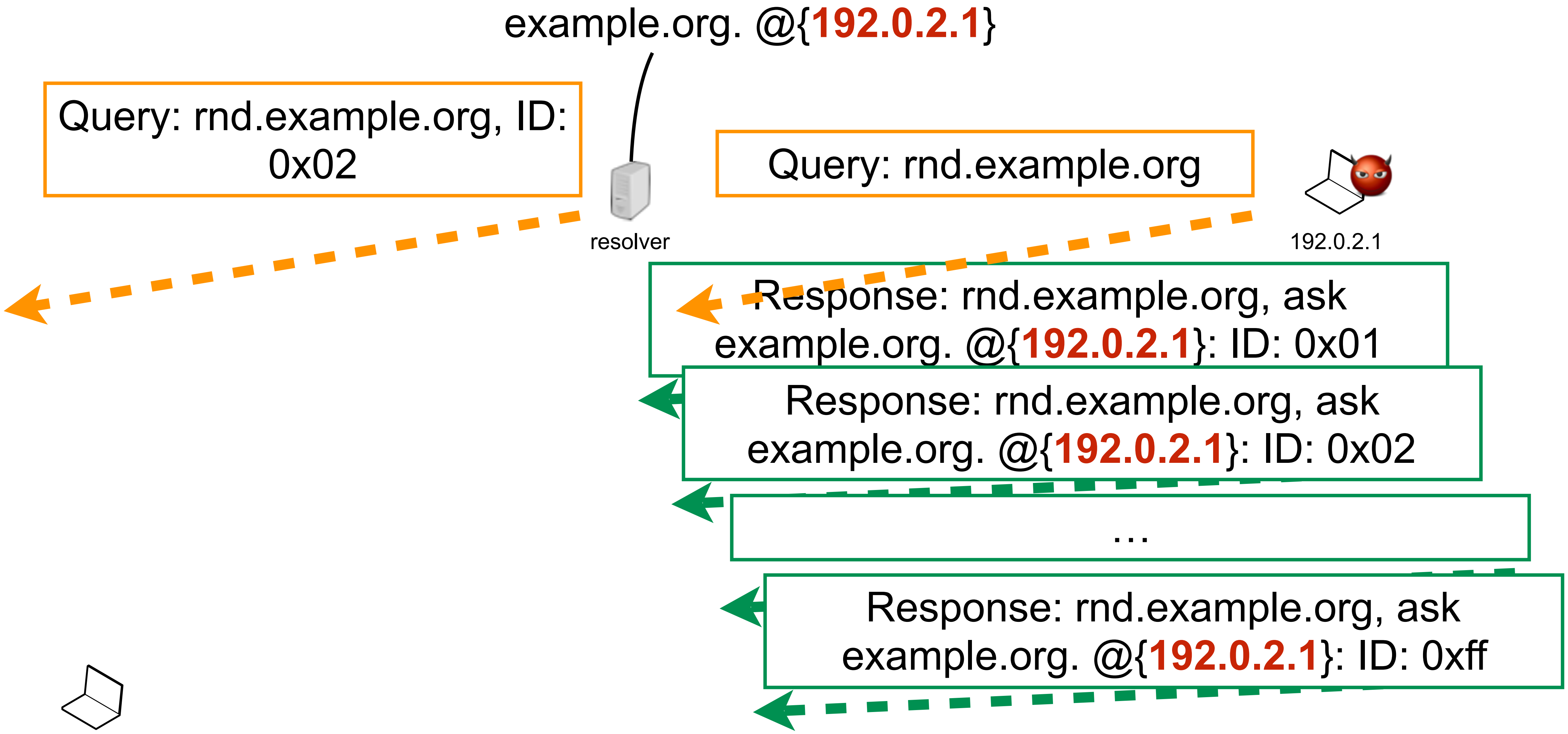
192.0.2.1

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x01

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02

…

DNS cache poisoning



example.org. @{192.0.2.1}

Query: rnd.example.org, ID: 0x02

Query: rnd.example.org

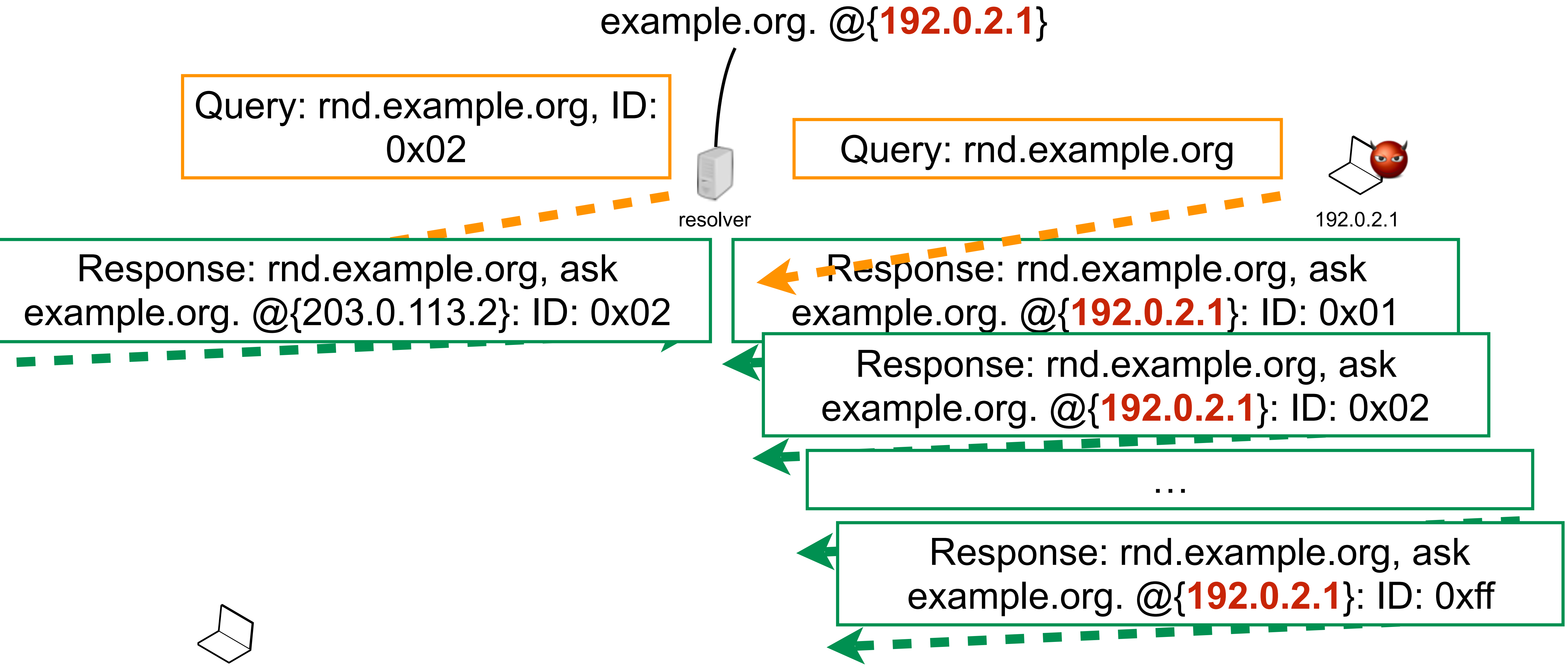
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x01

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02

…

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0xff

DNS cache poisoning



example.org. @{192.0.2.1}

Query: rnd.example.org, ID: 0x02

Query: rnd.example.org

Response: rnd.example.org, ask example.org. @{203.0.113.2}: ID: 0x02

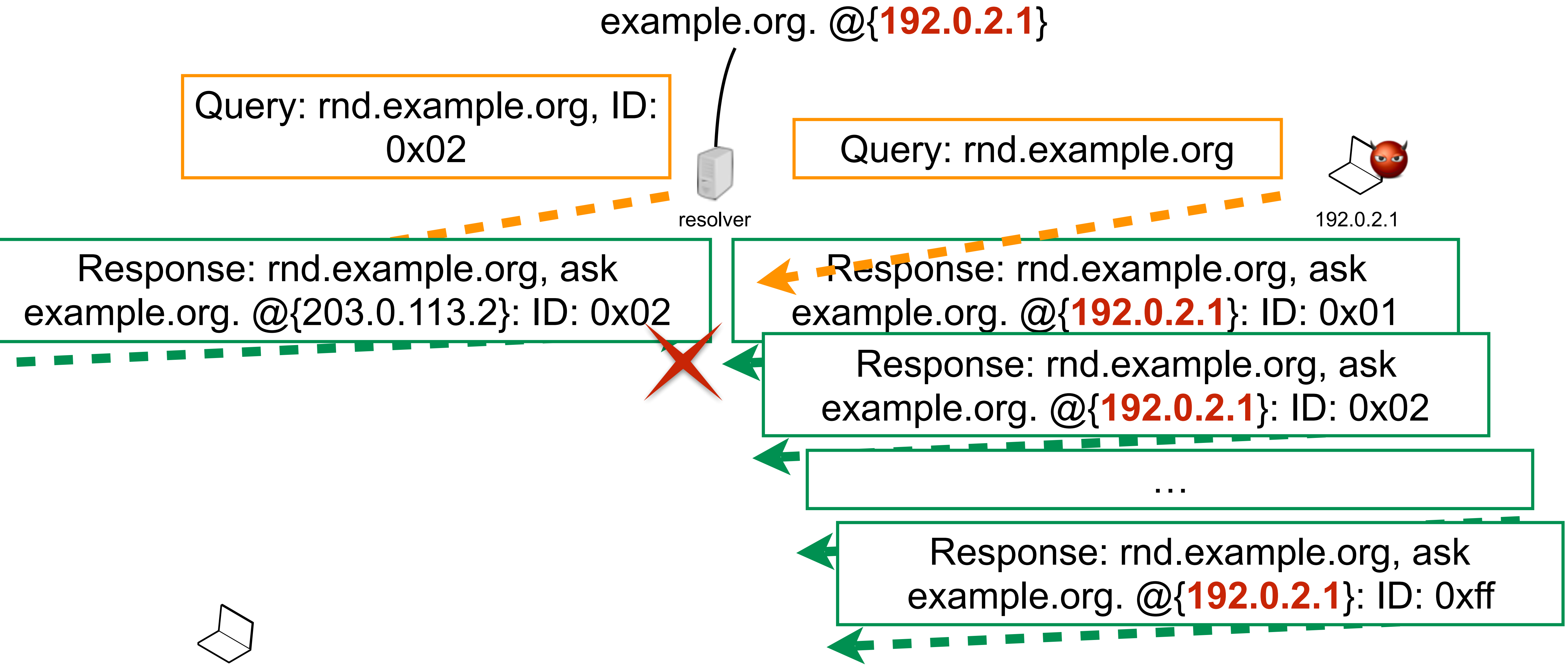
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x01

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02

…

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0xff

DNS cache poisoning



example.org. @{192.0.2.1}

Query: rnd.example.org, ID: 0x02

resolver

Query: rnd.example.org

192.0.2.1

Response: rnd.example.org, ask example.org. @{203.0.113.2}: ID: 0x02

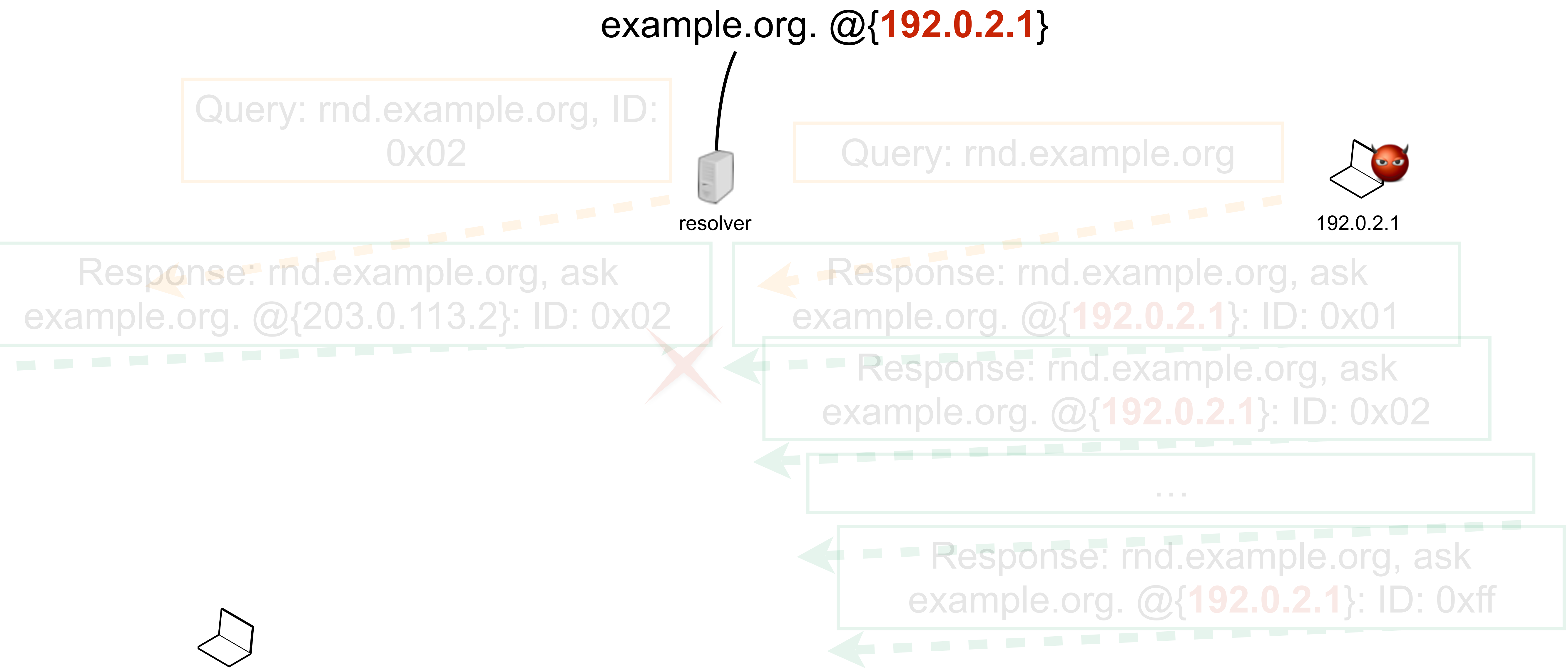
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x01

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02

…

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0xff

DNS cache poisoning



example.org. @{192.0.2.1}

Query: rnd.example.org, ID: 0x02

Query: rnd.example.org

resolver

192.0.2.1

Response: rnd.example.org, ask example.org. @{203.0.113.2}: ID: 0x02

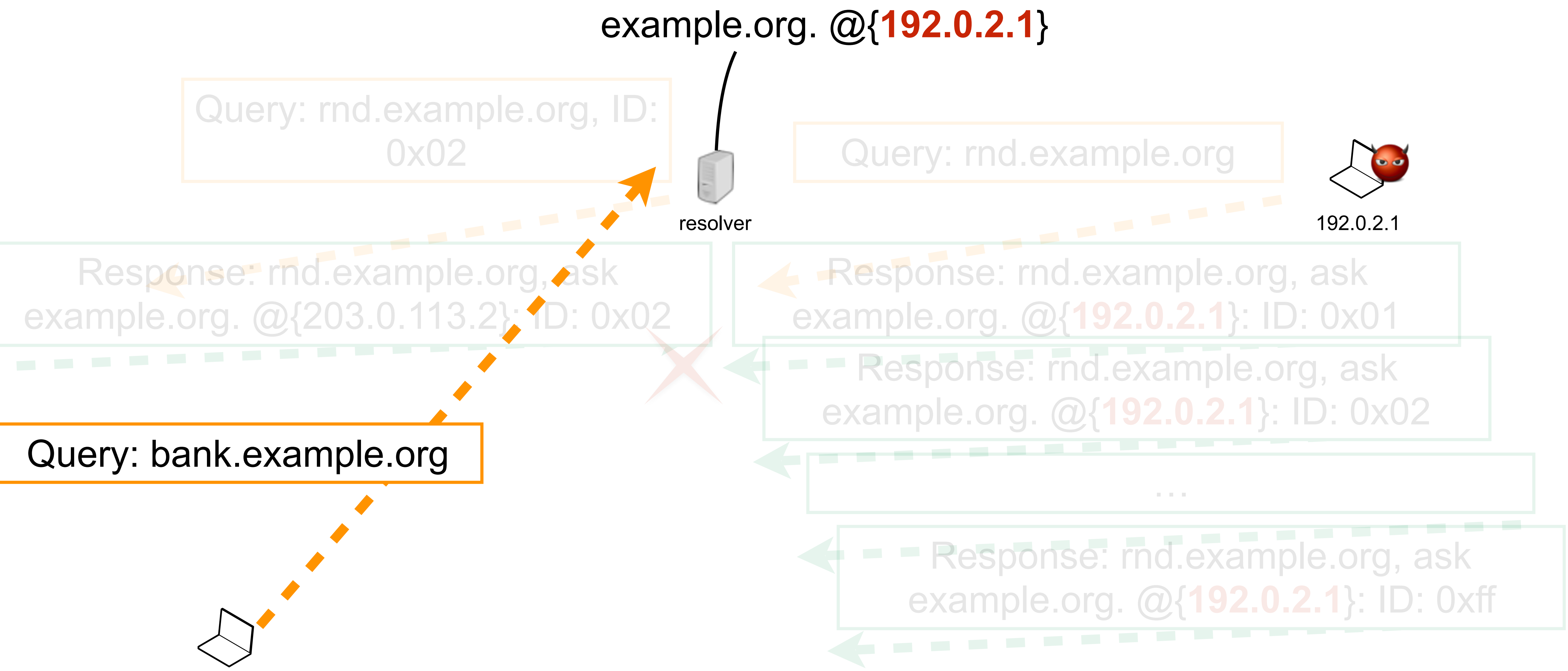
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x01

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02

…

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0xff

DNS cache poisoning



example.org. @{192.0.2.1}

Query: rnd.example.org, ID: 0x02

resolver

Query: rnd.example.org

192.0.2.1

Response: rnd.example.org, ask example.org. @{203.0.113.2}: ID: 0x02

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x01

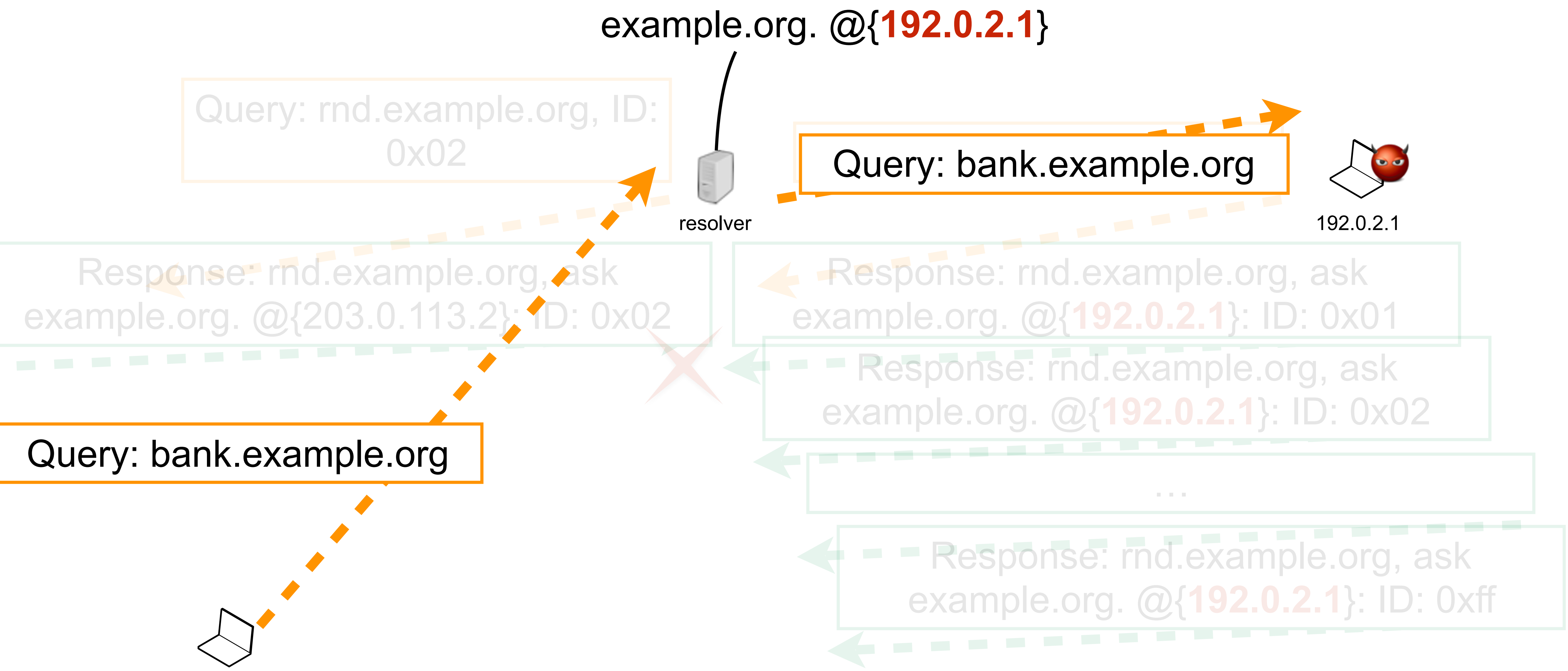
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02

Query: bank.example.org

…

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0xff

DNS cache poisoning



example.org. @{192.0.2.1}

Query: rnd.example.org, ID: 0x02

Query: bank.example.org

resolver

192.0.2.1

Response: rnd.example.org, ask example.org. @{203.0.113.2}: ID: 0x02

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x01

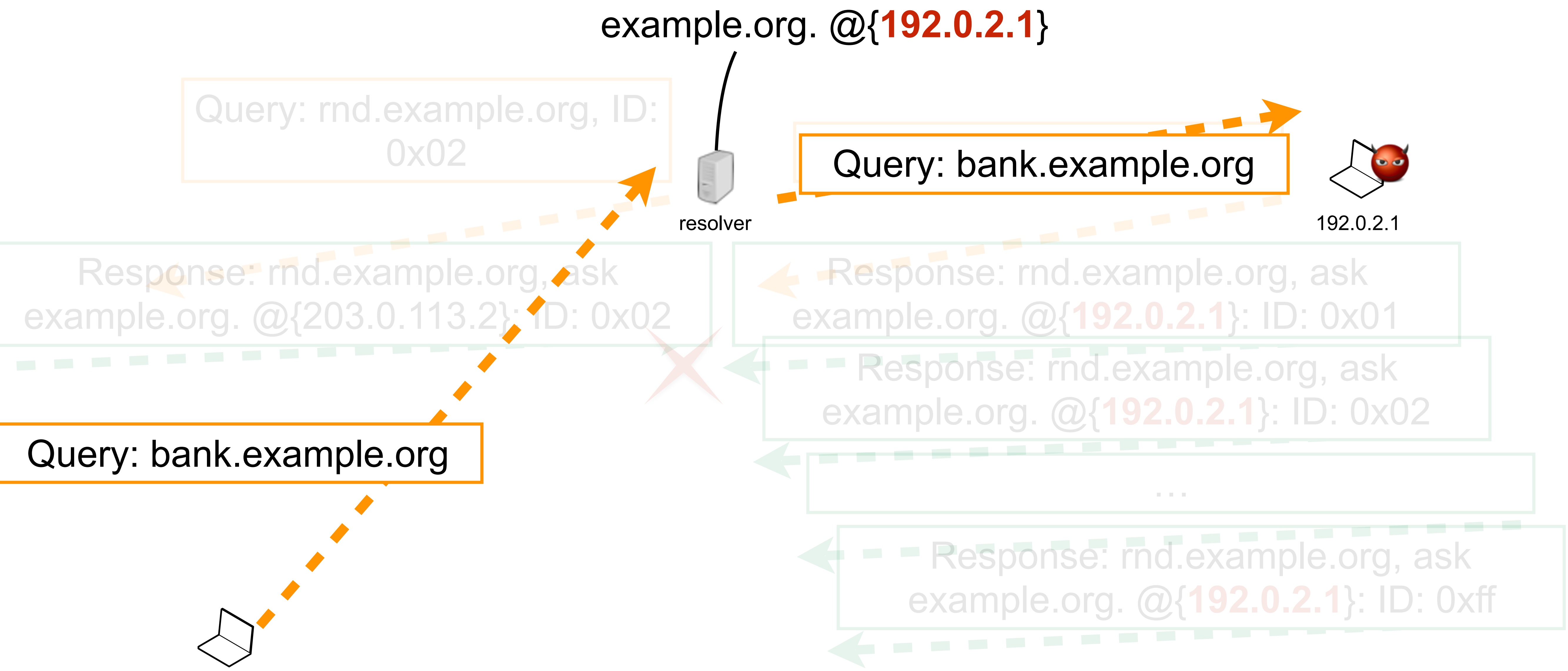
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02

…

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0xff

Query: bank.example.org

DNS cache poisoning

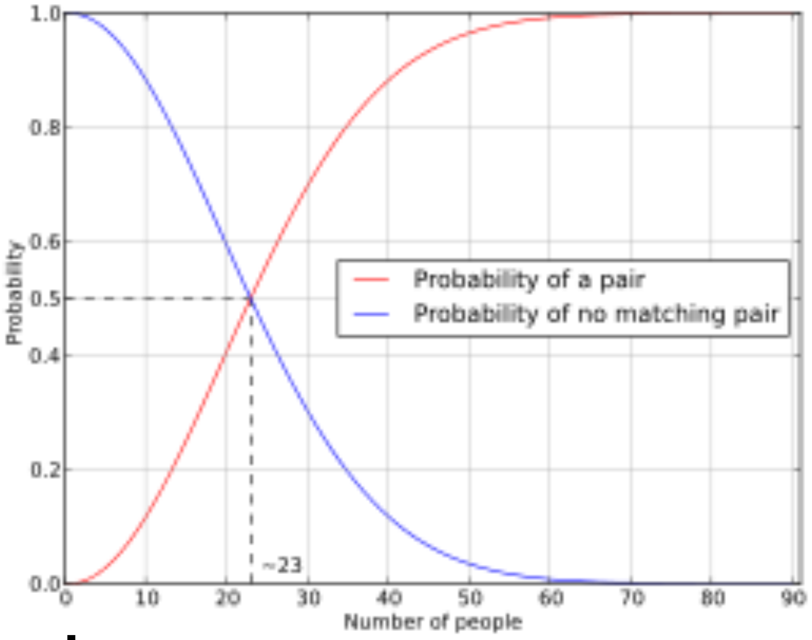

example.org. @{192.0.2.1}
Query: rnd.example.org, ID: 0x02
resolver
Query: bank.example.org
192.0.2.1
Response: rnd.example.org, ask example.org. @{203.0.113.2}: ID: 0x02
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x01
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02
…
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0xff
Query: bank.example.org

Why does it work?

Why does it work?

- Birthday paradox
  - probability that *n* elements uniformly picked from the finite set *T* is

$$p(n) = 1 - \frac{|T|!}{(|T| - n)} \cdot \frac{1}{|T|^n}$$



- Relying solely on transaction ID is dangerous
  - particularly when IDs are small (16 bits in DNS)

DNS Distributed Denial of Service (DDoS)

- Attacks against Dyn DNS infrastructure
- Two bursts: 2016-10-21 11:10 UTC - 13:20 UTC; 15:50 UTC - 20:30 UTC
- Not usual DDoS
  - many more addresses than usual, non spoofed (between 40k and 100k addresses)

https://www.ietf.org/proceedings/97/slides/slides-97-ietf-sessb-the-internets-architecture-is-under-attack-ironically-andrew-sullivan-00.pdf

Why does it work?

- Attacks performed via a Mirai-based botnet
  - IoT devices
- End-to-End principle
  - maximizes the intelligence at the edge
  - network avoids making decisions
- What if the edge is “bad”?

YouTube Hijacking

- *BBC Breaking news: A router problem made YouTube inaccessible for many*
- *RIPE NIS: “On Sunday, 24 February 2008, Pakistan Telecom (AS17557) started an unauthorised announcement of the prefix 208.65.153.0/24. One of Pakistan Telecom's upstream providers, PCCW Global (AS3491) forwarded this announcement to the rest of the Internet, which resulted in the hijacking of YouTube traffic on a global scale”*

http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study

YouTube Hijacking (contd.)

http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study

YouTube Hijacking (contd.)

- **Before, during and after Sunday, 24 February 2008**: AS36561 (YouTube) announces **208.65.152.0/22**.

http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study

YouTube Hijacking (contd.)

- **Before, during and after Sunday, 24 February 2008**: AS36561 (YouTube) announces **208.65.152.0/22**.
- **Sunday, 24 February 2008, 18:47 (UTC)**: AS17557 (Pakistan Telecom) starts announcing **208.65.153.0/24**. AS3491 (PCCW Global) propagates the announcement. Routers around the world receive the announcement, and YouTube traffic is redirected to Pakistan.

http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study

YouTube Hijacking (contd.)

- **Before, during and after Sunday, 24 February 2008**: AS36561 (YouTube) announces **208.65.152.0/22**.
- **Sunday, 24 February 2008, 18:47 (UTC)**: AS17557 (Pakistan Telecom) starts announcing **208.65.153.0/24**. AS3491 (PCCW Global) propagates the announcement. Routers around the world receive the announcement, and YouTube traffic is redirected to Pakistan.
- **Sunday, 24 February 2008, 20:07 (UTC):** AS36561 (YouTube) starts announcing **208.65.153.0/24**. [...] BGP decision process means that AS17557 (Pakistan Telecom) continues to attract some of YouTube's traffic.

http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study

YouTube Hijacking (contd.)

- **Before, during and after Sunday, 24 February 2008**: AS36561 (YouTube) announces **208.65.152.0/22**.
- **Sunday, 24 February 2008, 18:47 (UTC)**: AS17557 (Pakistan Telecom) starts announcing **208.65.153.0/24**. AS3491 (PCCW Global) propagates the announcement. Routers around the world receive the announcement, and YouTube traffic is redirected to Pakistan.
- **Sunday, 24 February 2008, 20:07 (UTC):** AS36561 (YouTube) starts announcing **208.65.153.0/24**. [...] BGP decision process means that AS17557 (Pakistan Telecom) continues to attract some of YouTube's traffic.
- **Sunday, 24 February 2008, 20:18 (UTC):** AS36561 (YouTube) starts announcing **208.65.153.128/25** and **208.65.153.0/25**. Because of the longest prefix match rule, every router that receives these announcements will send the traffic to YouTube.

http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study

YouTube Hijacking (contd.)

- **Before, during and after Sunday, 24 February 2008**: AS36561 (YouTube) announces **208.65.152.0/22**.
- **Sunday, 24 February 2008, 18:47 (UTC)**: AS17557 (Pakistan Telecom) starts announcing **208.65.153.0/24**. AS3491 (PCCW Global) propagates the announcement. Routers around the world receive the announcement, and YouTube traffic is redirected to Pakistan.
- **Sunday, 24 February 2008, 20:07 (UTC):** AS36561 (YouTube) starts announcing **208.65.153.0/24**. [...] BGP decision process means that AS17557 (Pakistan Telecom) continues to attract some of YouTube's traffic.
- **Sunday, 24 February 2008, 20:18 (UTC):** AS36561 (YouTube) starts announcing **208.65.153.128/25** and **208.65.153.0/25**. Because of the longest prefix match rule, every router that receives these announcements will send the traffic to YouTube.
- **Sunday, 24 February 2008, 20:51 (UTC):** All prefix announcements, including the hijacked /24 which was originated by AS17557 (Pakistan Telecom) via AS3491 (PCCW Global), are seen **prepended by another 17557**. The longer AS path means that more routers prefer the announcement originated by YouTube.

http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study

YouTube Hijacking (contd.)

- **Before, during and after Sunday, 24 February 2008**: AS36561 (YouTube) announces **208.65.152.0/22**.
- **Sunday, 24 February 2008, 18:47 (UTC)**: AS17557 (Pakistan Telecom) starts announcing **208.65.153.0/24**. AS3491 (PCCW Global) propagates the announcement. Routers around the world receive the announcement, and YouTube traffic is redirected to Pakistan.
- **Sunday, 24 February 2008, 20:07 (UTC):** AS36561 (YouTube) starts announcing **208.65.153.0/24**. [...] BGP decision process means that AS17557 (Pakistan Telecom) continues to attract some of YouTube's traffic.
- **Sunday, 24 February 2008, 20:18 (UTC):** AS36561 (YouTube) starts announcing **208.65.153.128/25** and **208.65.153.0/25**. Because of the longest prefix match rule, every router that receives these announcements will send the traffic to YouTube.
- **Sunday, 24 February 2008, 20:51 (UTC):** All prefix announcements, including the hijacked /24 which was originated by AS17557 (Pakistan Telecom) via AS3491 (PCCW Global), are seen **prepended by another 17557**. The longer AS path means that more routers prefer the announcement originated by YouTube.
- **Sunday, 24 February 2008, 21:01 (UTC):** AS3491 (PCCW Global) **withdraws all prefixes originated by AS17557** (Pakistan Telecom), thus stopping the hijack of 208.65.153.0/24. Note that AS17557 was not completely disconnected by AS3491. Prefixes originated by other Pakistani ASs were still announced by AS17557 through AS3491.

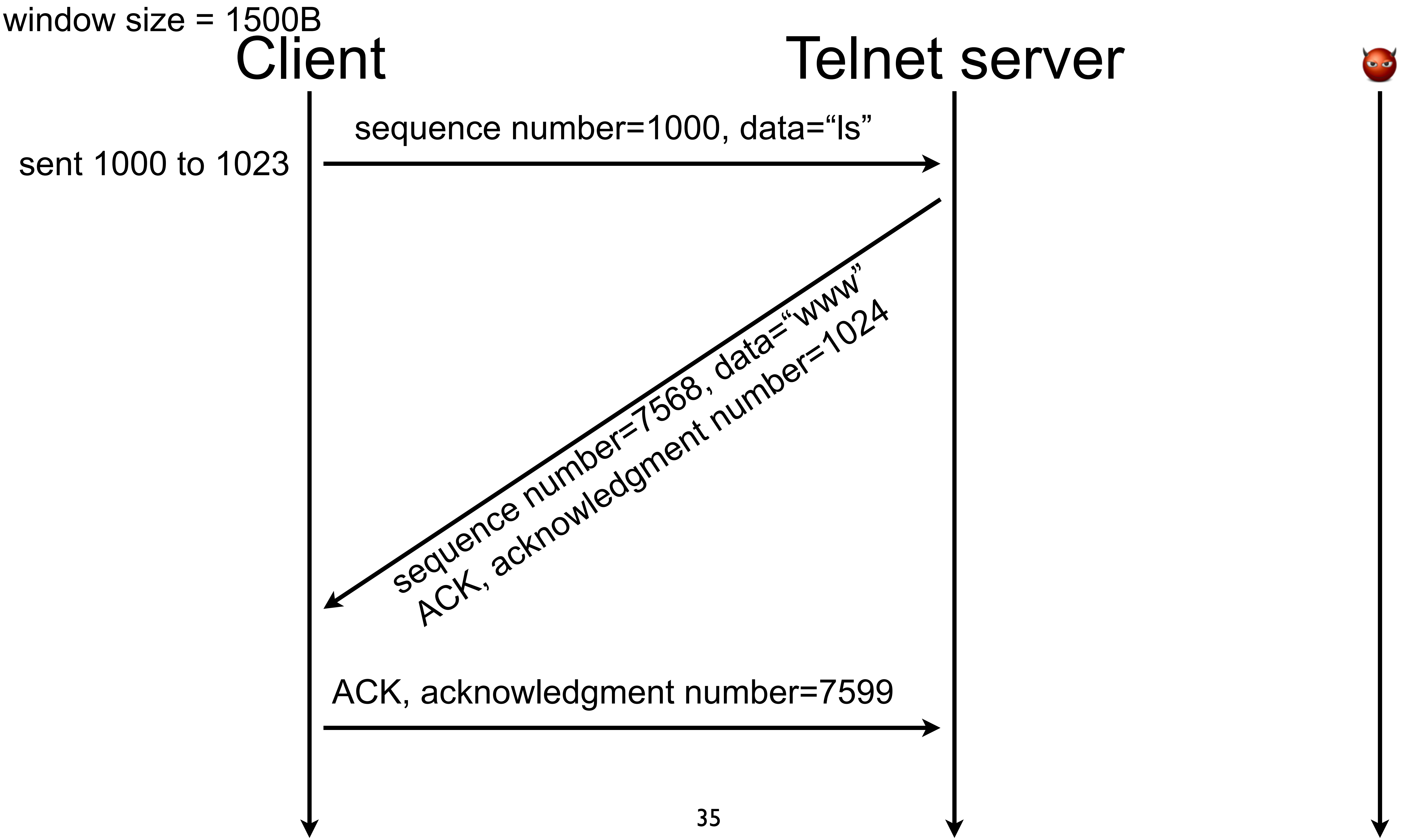
http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study

Why does it work?

Why does it work?

- Any AS can claim to be the originator of a prefix (i.e., she hijacks the prefix)
- To protect against that, only the import filters can be used
  - rely on databases that are not so accurate
- A not secure global routing system is a major threat against freedom

TCP session hijacking



window size = 1500B

Client

Telnet server

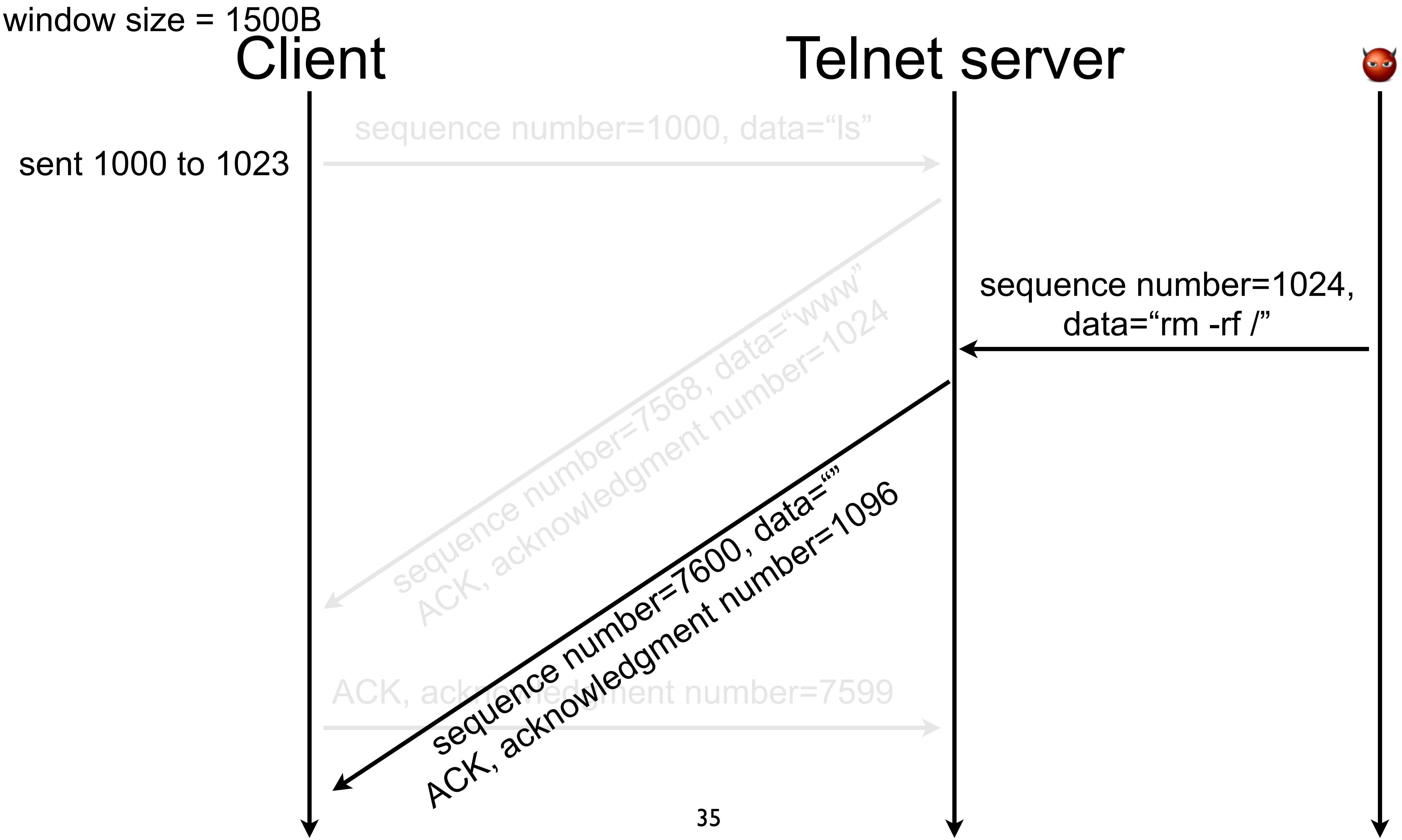
sent 1000 to 1023

sequence number=1000, data="ls"

sequence number=7568, data="www"
ACK, acknowledgment number=1024

ACK, acknowledgment number=7599

TCP session hijacking



window size = 1500B

Client

Telnet server

sequence number=1000, data=“ls”

sent 1000 to 1023

sequence number=7568, data=“www”
ACK, acknowledgment number=1024

sequence number=1024,
data=“rm -rf /”

sequence number=7600, data=“”
ACK, acknowledgment number=1096

ACK, acknowledgment number=7599

Why does it work?

Why does it work?

- If the attacker can
  - guess the initial sequence number
  - guess actions from the sender
- then easy to guess a sequence number that will be accepted by the receiver

The basics of security

Security threats

- Intrusion
  - an attacker gains remote access to some resources that are normally denied to her
    - e.g., steal processing power, botnets
- Eavesdropping
  - an attacker collects traffic of a target in order to gain access to restricted sensitive information
    - e.g., steal passwords by sniffing wireless traffic
- Denial of Service (DoS)
  - an attacker disrupts a specific targeted service
    - e.g., block the youtube website

The attackers

- Hackers
  - look for challenge, notoriety, and fun
    - e.g., hackers, script kiddies, students :-D
- Spies
  - look for political/business gains
    - e.g., intelligence, police, industrial spies
- Criminals
  - look for financial gains, religious/political visibility, or just to break something
  - e.g., criminals, terrorists, vandals

Definitions

- Key
  - input of cryptographic functions to determine its output
- Authentication
  - proof that the message is coming from the one claiming to be at the origin of the message
- Integrity
  - proof that the message has not been altered since its creation
- Non-repudiation of origin
  - an entity that generated a message cannot deny have generated the message
- Encryption
  - action of encoding of a message such that an eavesdropper can’t read the message but legitimate destination can
- Decryption
  - action of decoding an encrypted message
- Signature
  - a mathematically constructed proof of authenticity of a message

Hall of fame

- Alice and Bob
  - are legitimate users, Alice and Bob exchange messages
- Chuck
  - is a malicious user that is not between Alice and Bob
- Eve
  - is a malicious user that can eavesdrop
- Trudy
  - is a malicious user that can perform (wo)man-in-the-middle attacks
- Trent
  - is a legitimate user that plays the role of a trusted arbitrator

Why is good security level so hard to obtain?

- The security level of a system equals the security level of the weakest part of the system
  - e.g., encrypting your HDD to avoid information leak if the laptop is stollen is useless if the password is written on a post-it attached on the laptop
- Digital system are complexes
  - interactions with many components, distribution, easily bugged...

Security is a tradeoff

- Compare cost and probability of an attack and cost of securing the system against this attack
  - e.g., is that necessary to make data unbreakable for 20 years if they are outdated after 1 hour?
- Explain the security systems and their reasons
  - if a user does not understand why he must follow a procedure, he will not follow it
    - e.g., how many of you already give their password to someone else?
- Never “over-secure” a system
  - if the system is too hard to use, people will find countermeasure
    - e.g., too hard to use corporate mails? Then use gmail to send corporate mails...

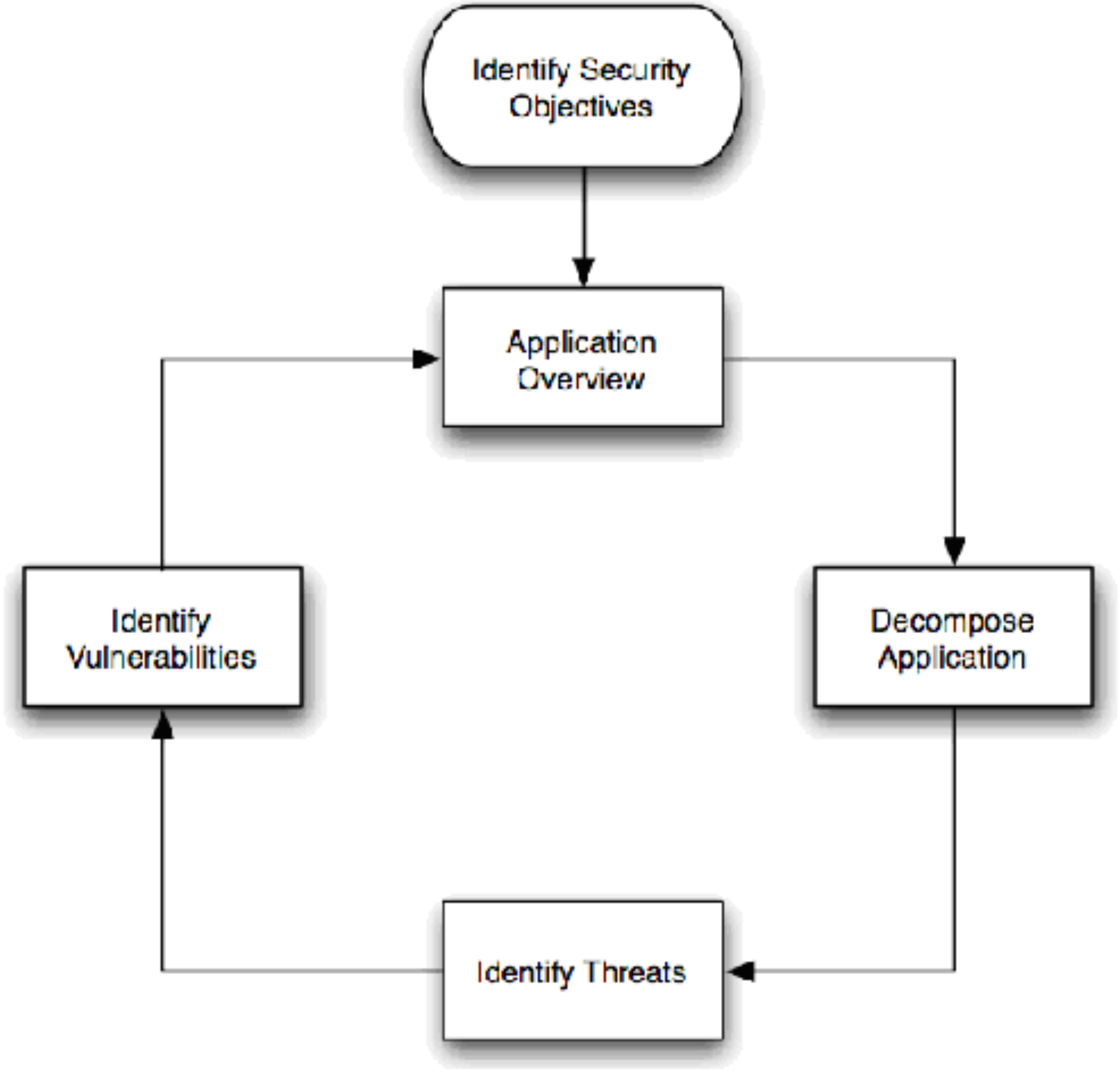
Security is a tradeoff (contd.)

- Protection system
  - lifetime = 10 years
  - cost = 10,000 EUR
- Attack
  - yearly probability = 10%
  - cost of restoring the system = 1,000 EUR
- Do I invest?

Procedures!

- Protection will never be perfect
- Prepare procedures
  - what to do BEFORE an attack?
    - what to do to limit the risk (e.g., passwords) of attack and to be ready if an attack happens (e.g., backup)
  - what to do DURING an attack?
    - the attack is on going, how to stop it
  - what to do AFTER an attack?
    - the attack succeeded, how to recover from it

Threat Risk Modelling*



* https://www.owasp.org/index.php/Threat_Risk_Modeling

DREAD*

- **D**amage **R**eproducibility **E**xploitability **A**ffected users **D**iscoverability (DREAD) is a classification scheme to assess and compare the risk presented by each evaluated threat.
- Risk_DREAD = (DAMAGE + REPRODUCIBILITY + EXPLOITABILITY + AFFECTED USERS + DISCOVERABILITY) / 5
- Damage Potential (how much damage can it cause?)
  - e.g., 0 = nothing, 5 = some, 10 = complete
- Reproducibility (how easy is it to reproduce the threat?)
  - e.g., 0 = impossible, 5 = few steps, need authentication, 10 = simple, no authentication needed.
- Exploitability (what is needed to exploit this threat?)
  - e.g., 0 = advanced tools and knowledge, 4 = using public attack tools, 10 = just a web browser
- Affected users (how many users will be affected?)
  - e.g., 0 = none, 5 = some, 10 = all users
- Discoverability (how easy is it to discover this threat?)
  - e.g., 0 = very hard, 5 = need monitoring, 9 = documented publicly, 10 = visible in the address bar.

* https://www.owasp.org/index.php/Threat_Risk_Modeling

Securing communications

Objective

- Construct a communication mechanism where Alice and Bob can exchange messages such that
  - only Alice and Bob can generate messages
  - nobody else than Alice or Bob can read messages
  - nobody can alter messages

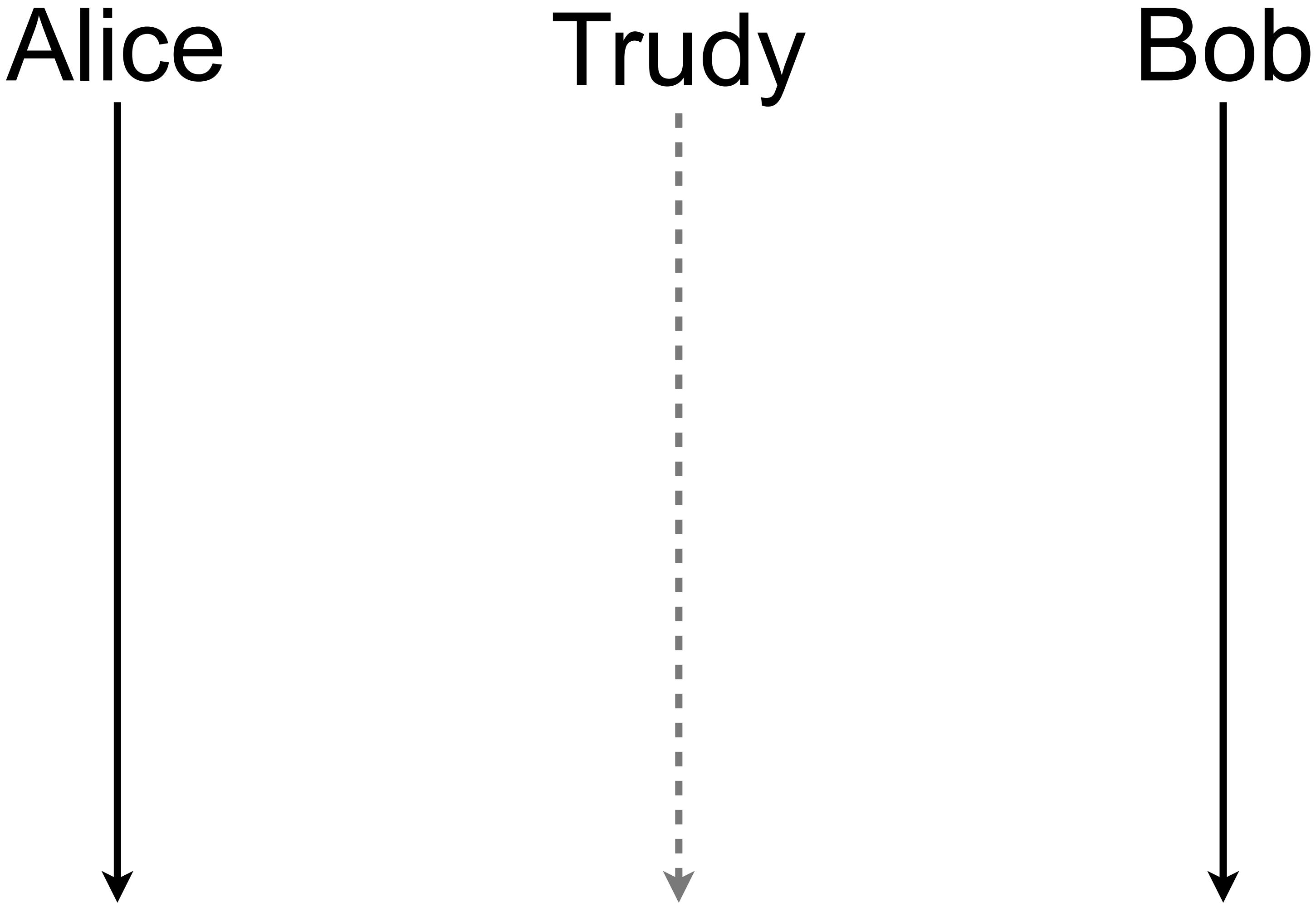
Steps

- fill me
- fill me
- fill me

Hash function

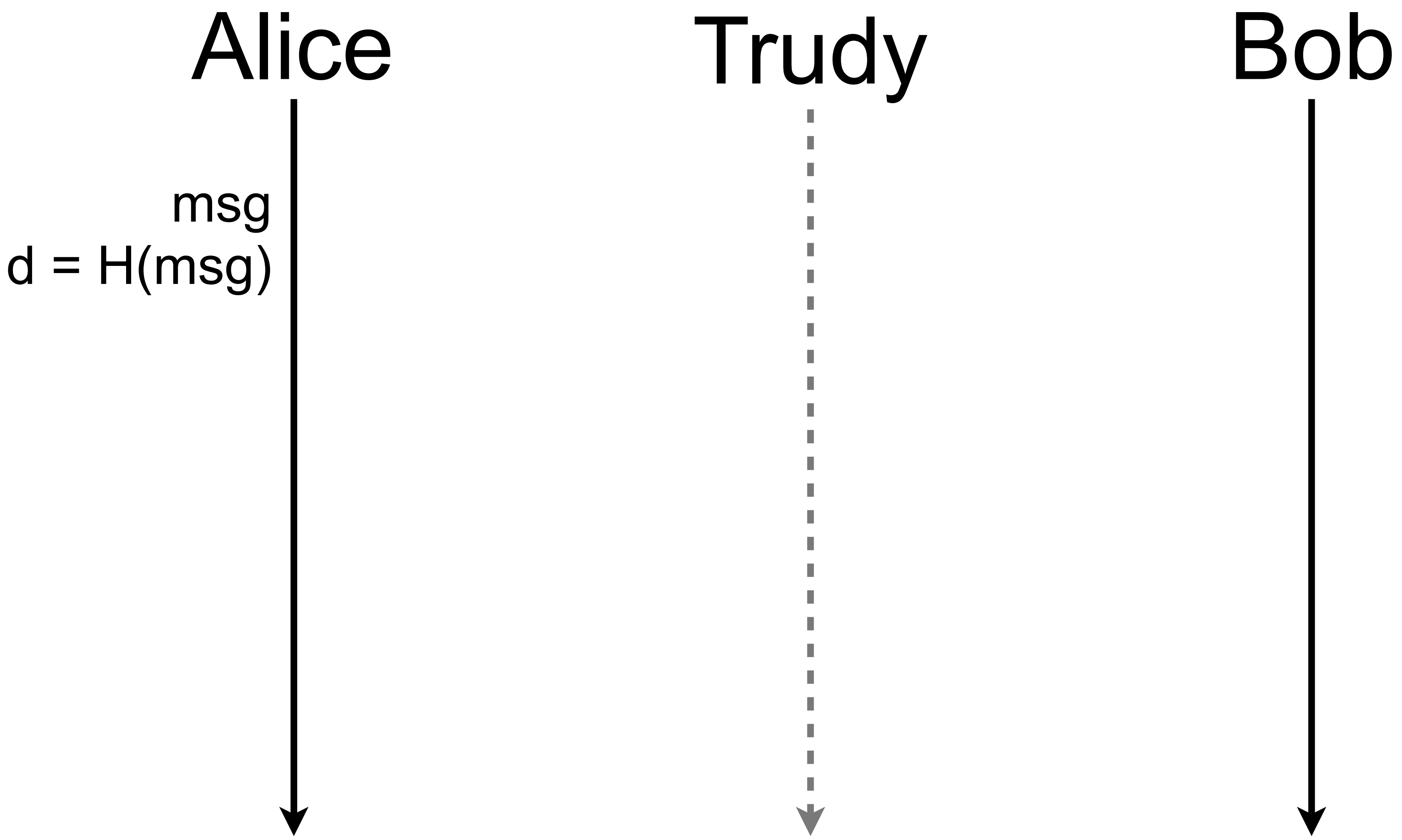
- Validate that a message has not been altered on its way between Alice and Bob
- Hash functions map arbitrary large numbers of variable length to fixed-length numbers
  - h = H(m), h is called hash or digest
  - e.g., MD5, SHA-1, SHA-256
- Good hash functions for cryptography must be such that
  - H(m) is not complex to compute
  - but finding a m_2 such that $H(m_2) = H(m)$ is complex,
  - H(m) is deterministic,
  - H output must be evenly distributed over the output set
- Example
  - SHA-1 maps messages its input space on a 160-bits output
    - SHA-1(Message to validate) = 5e06ee754bda0d33cf65ec305ffc779404e66029
    - SHA-1(Message t**O** validate) = b1c306f8cb792fa14d4d1fdcf6f37d86c2fe6bb9

Is that enough?

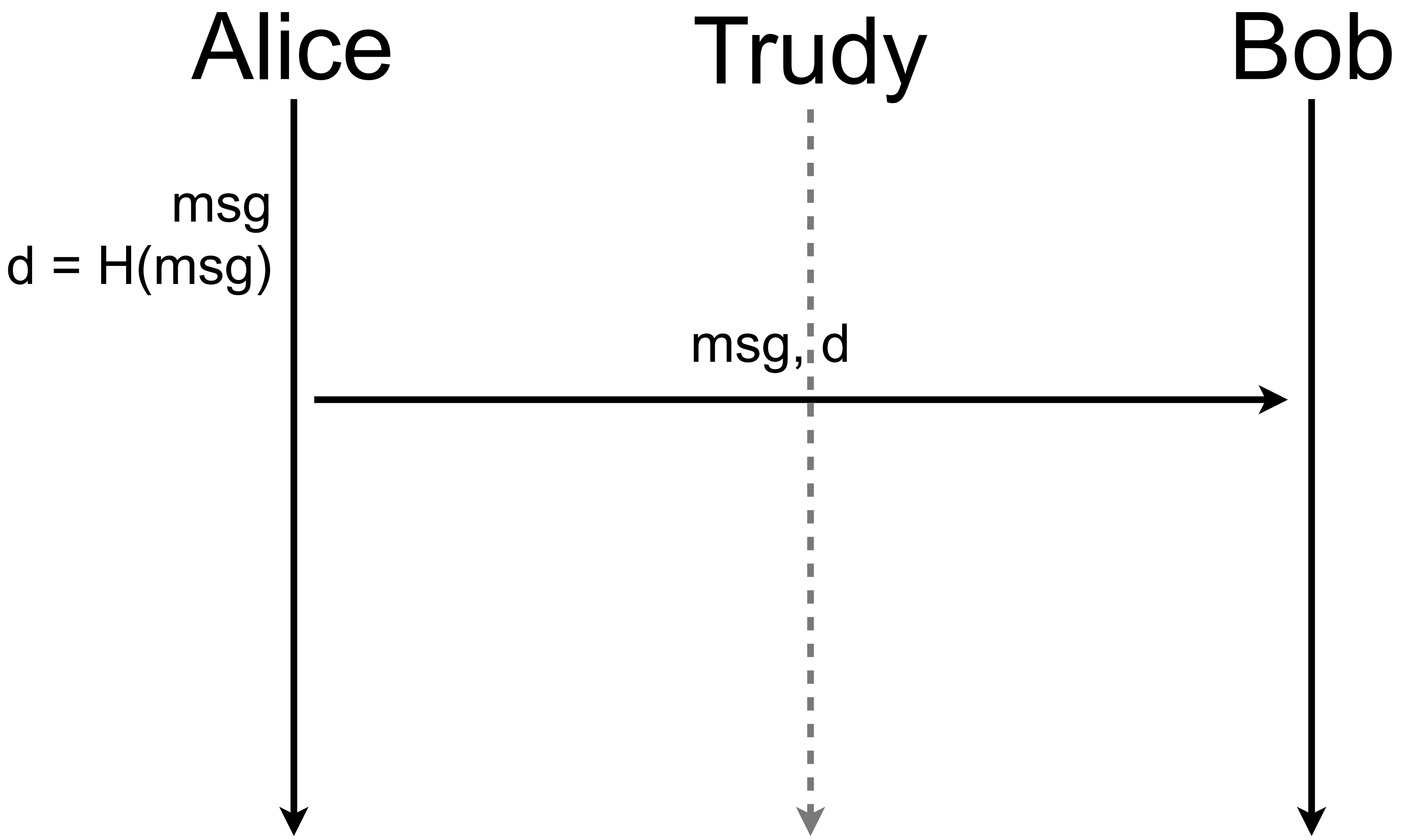


Alice Trudy Bob

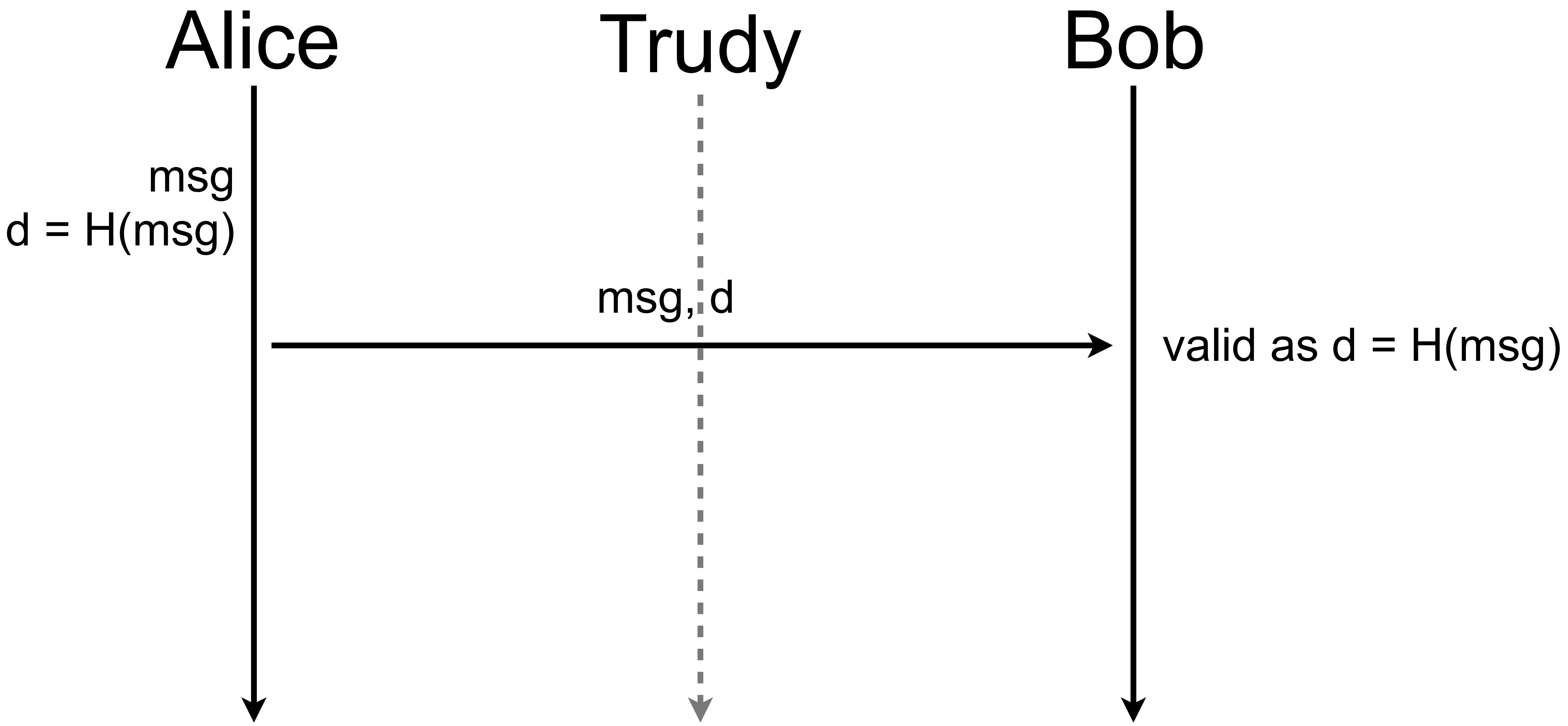
Is that enough?



Is that enough?



Is that enough?



Alice

Trudy

Bob

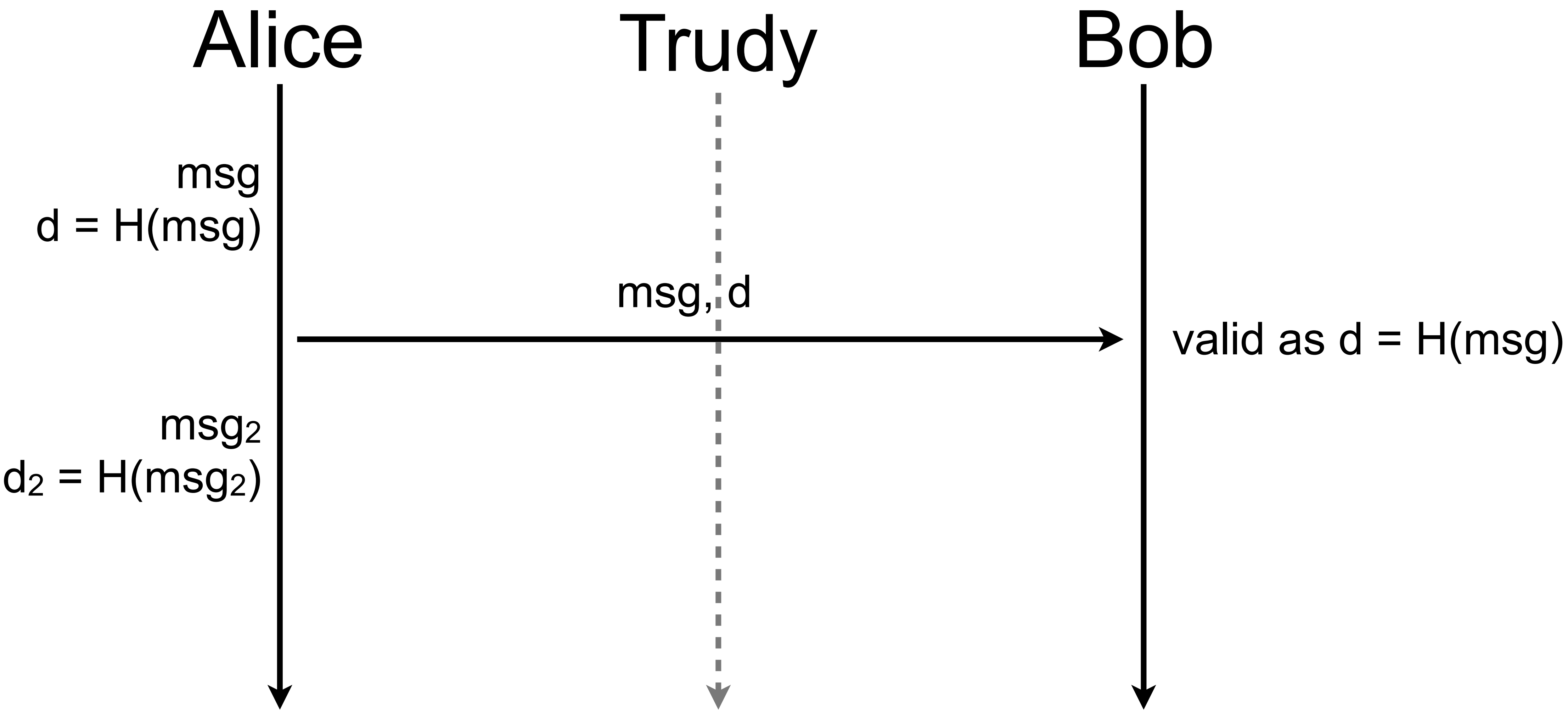
msg

d = H(msg)

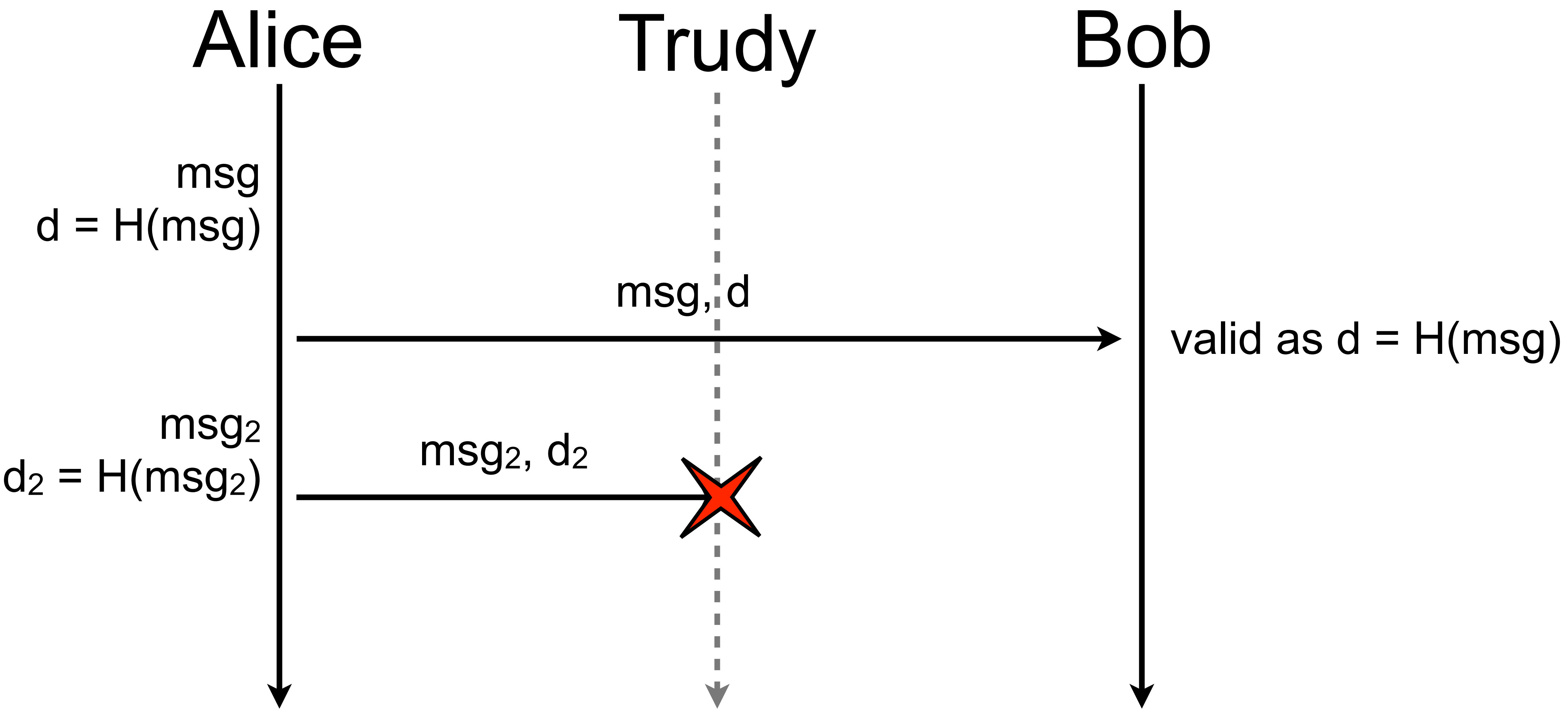
msg, d

valid as d = H(msg)

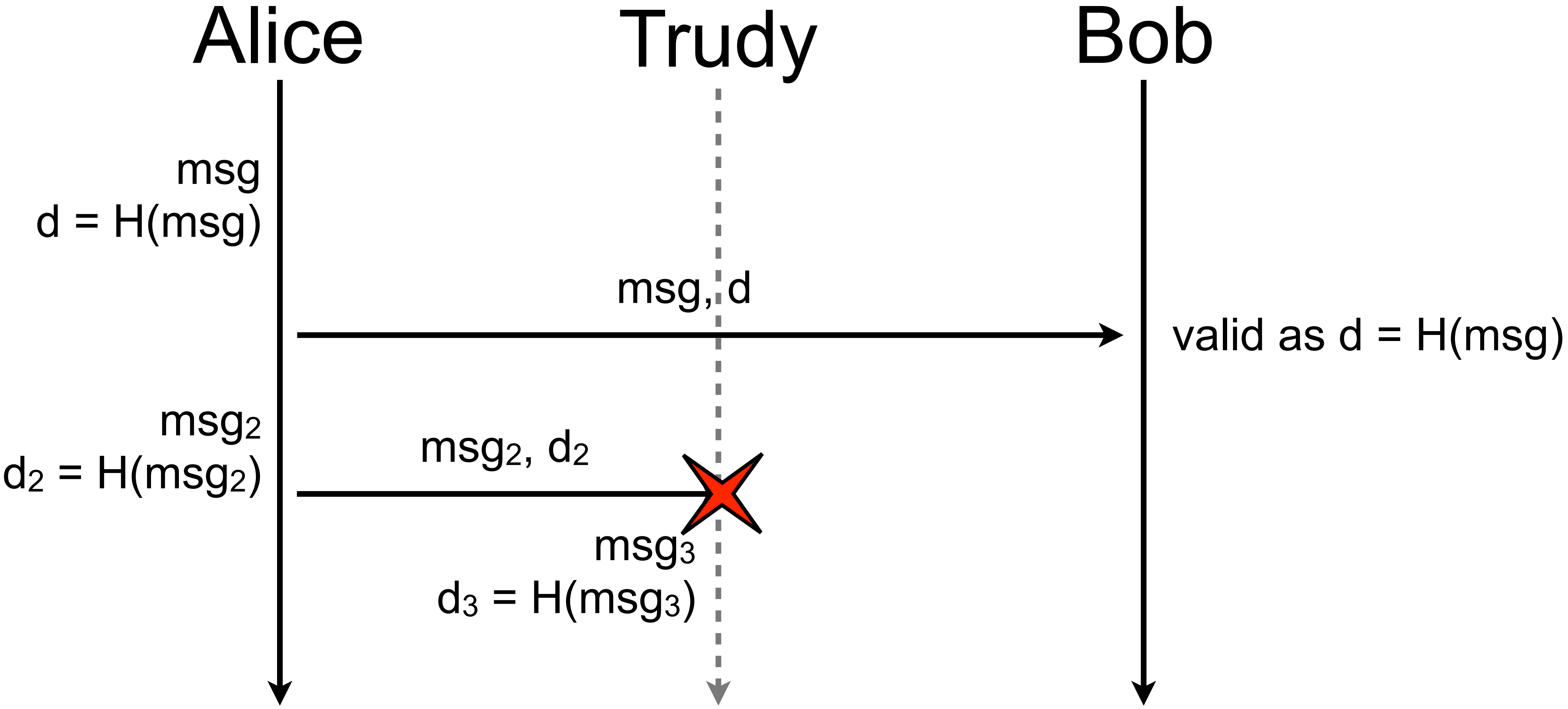
Is that enough?



Is that enough?



Is that enough?



Alice

Trudy

Bob

msg

d = H(msg)

msg, d

valid as d = H(msg)

msg_2

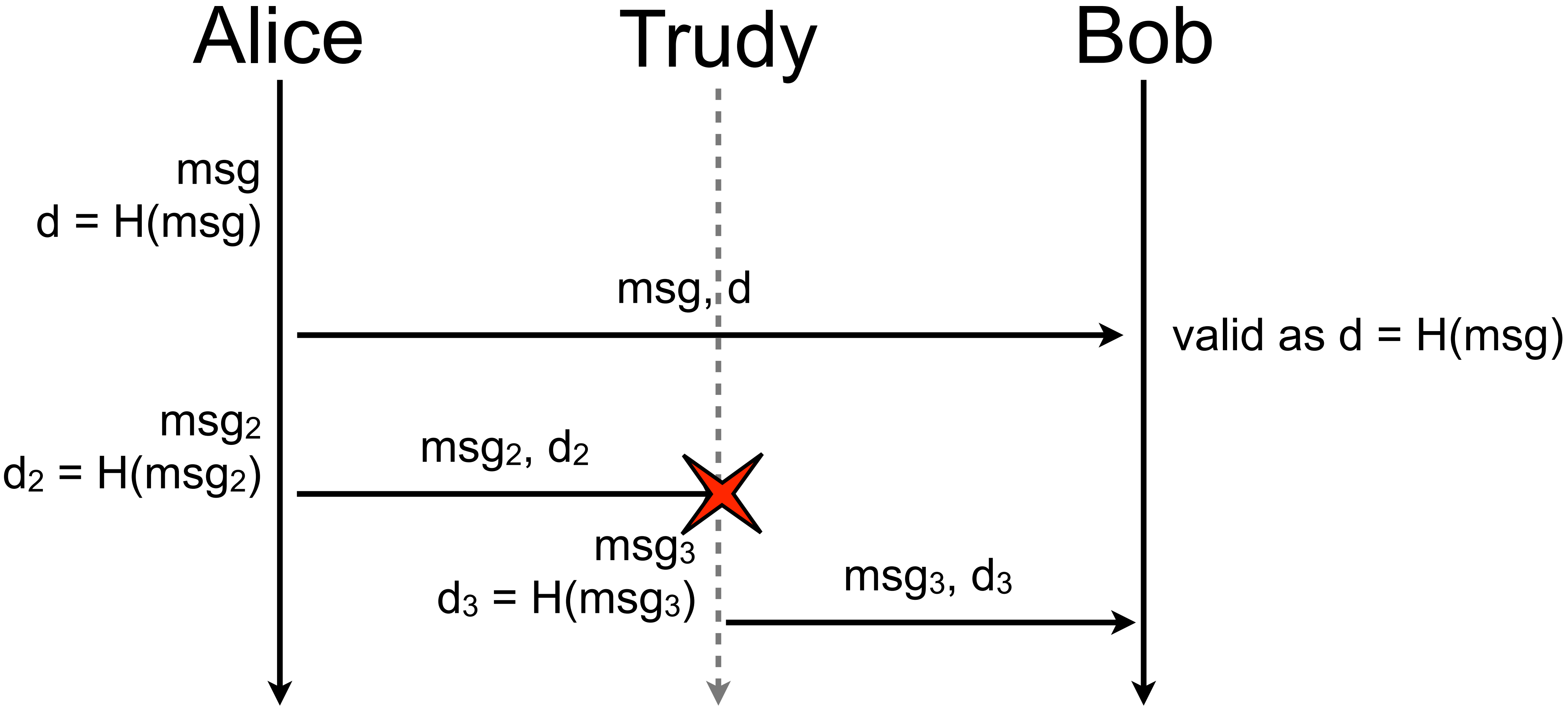
$d_2 = H(msg_2)$

msg_2, d_2

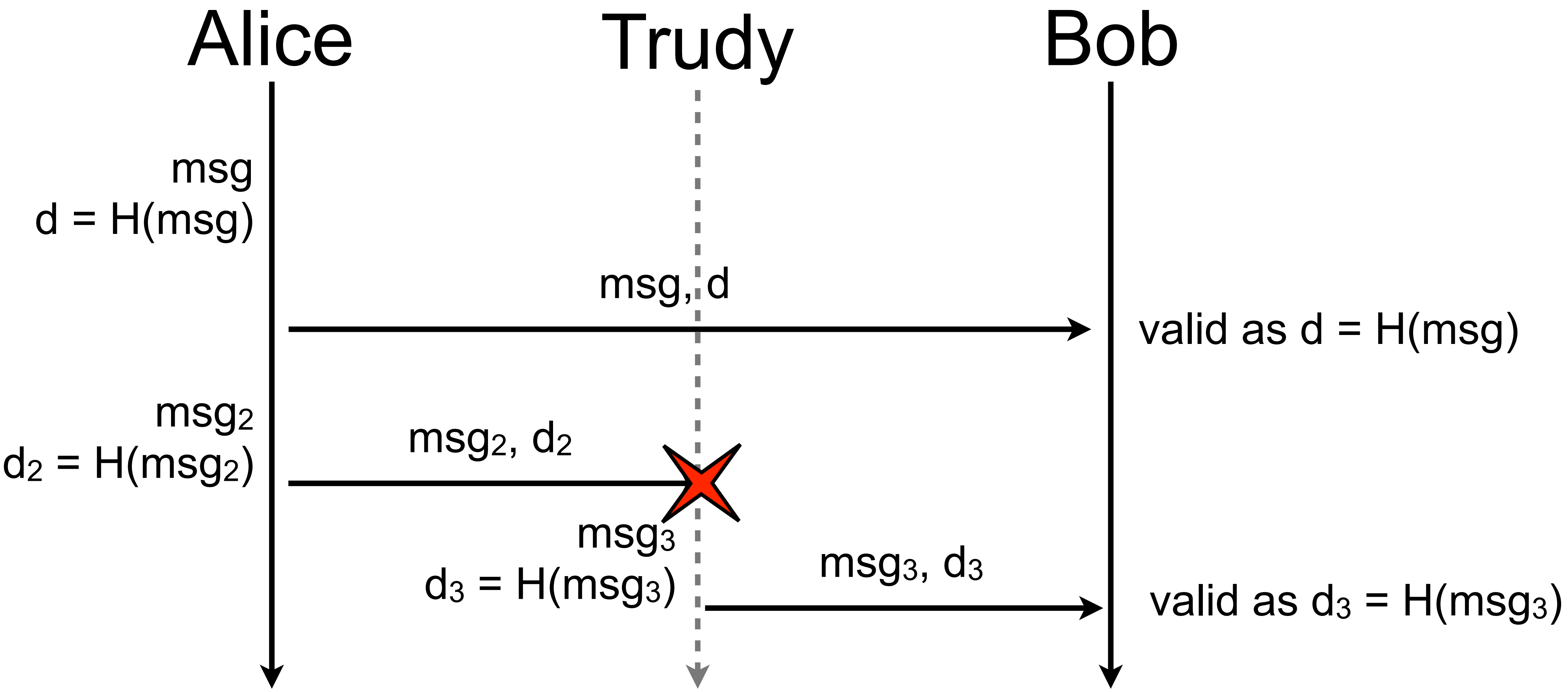
msg_3

$d_3 = H(msg_3)$

Is that enough?



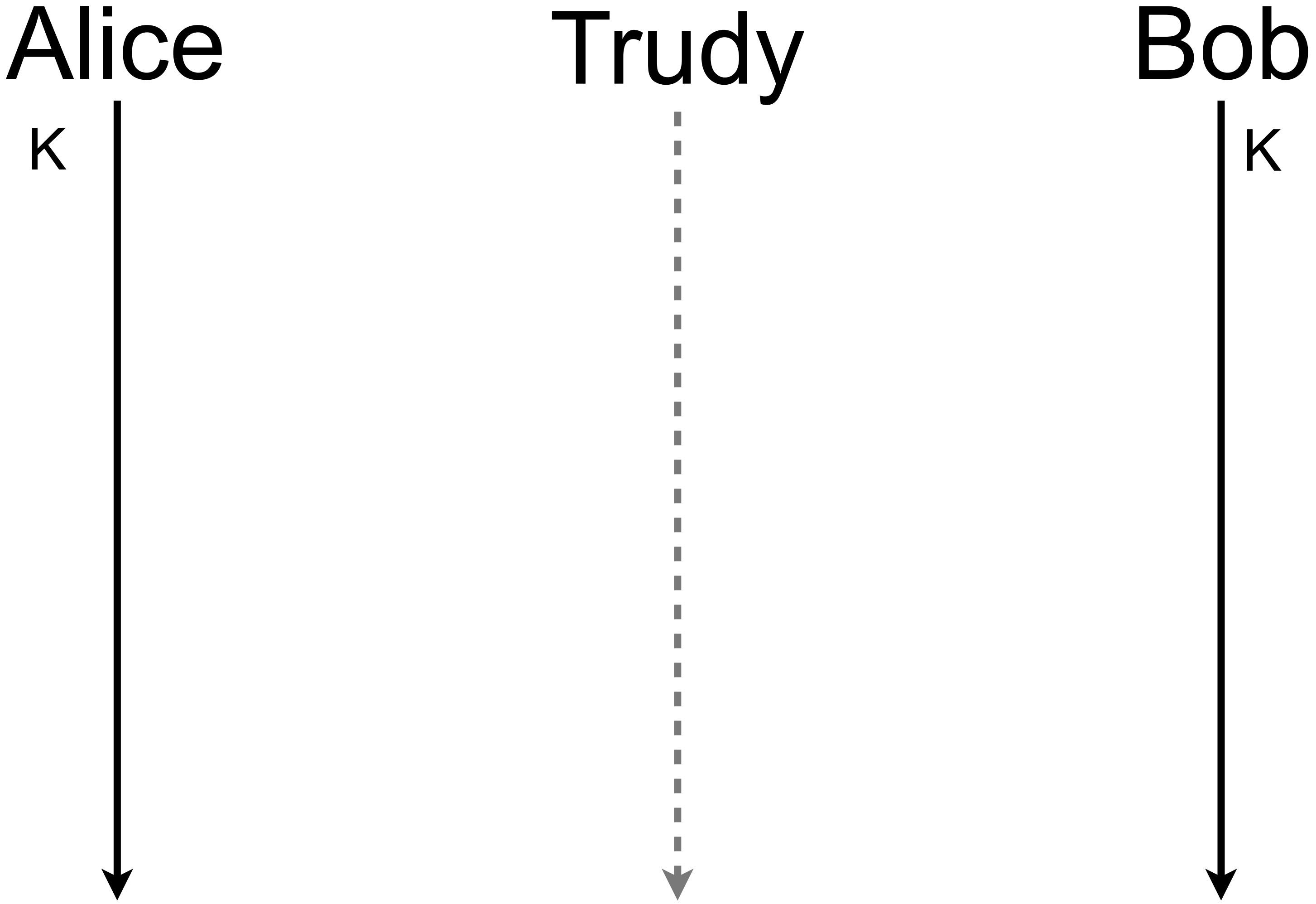
Is that enough?



Hash function with salt

- Hash functions are deterministic
- Add a salt such that the output of the hash function is a function of the message and the salt
    - h = H(m, K) where K is the salt or key of the hash function
- As long as Trudy does not know the salt, she can't forge a valid digest

Hash function with salt (contd.)



Alice

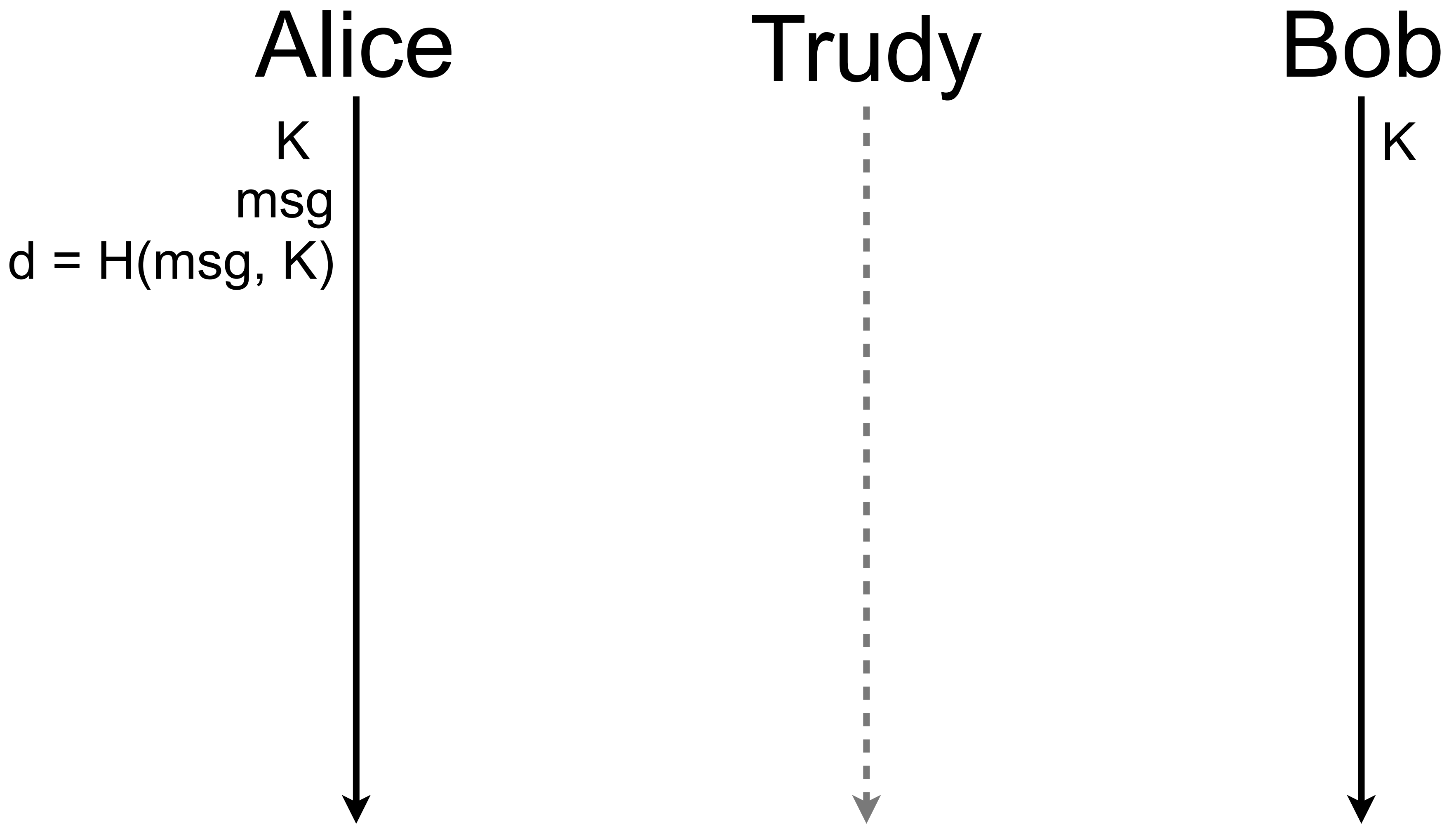
Trudy

Bob

K

K

Hash function with salt (contd.)



Alice

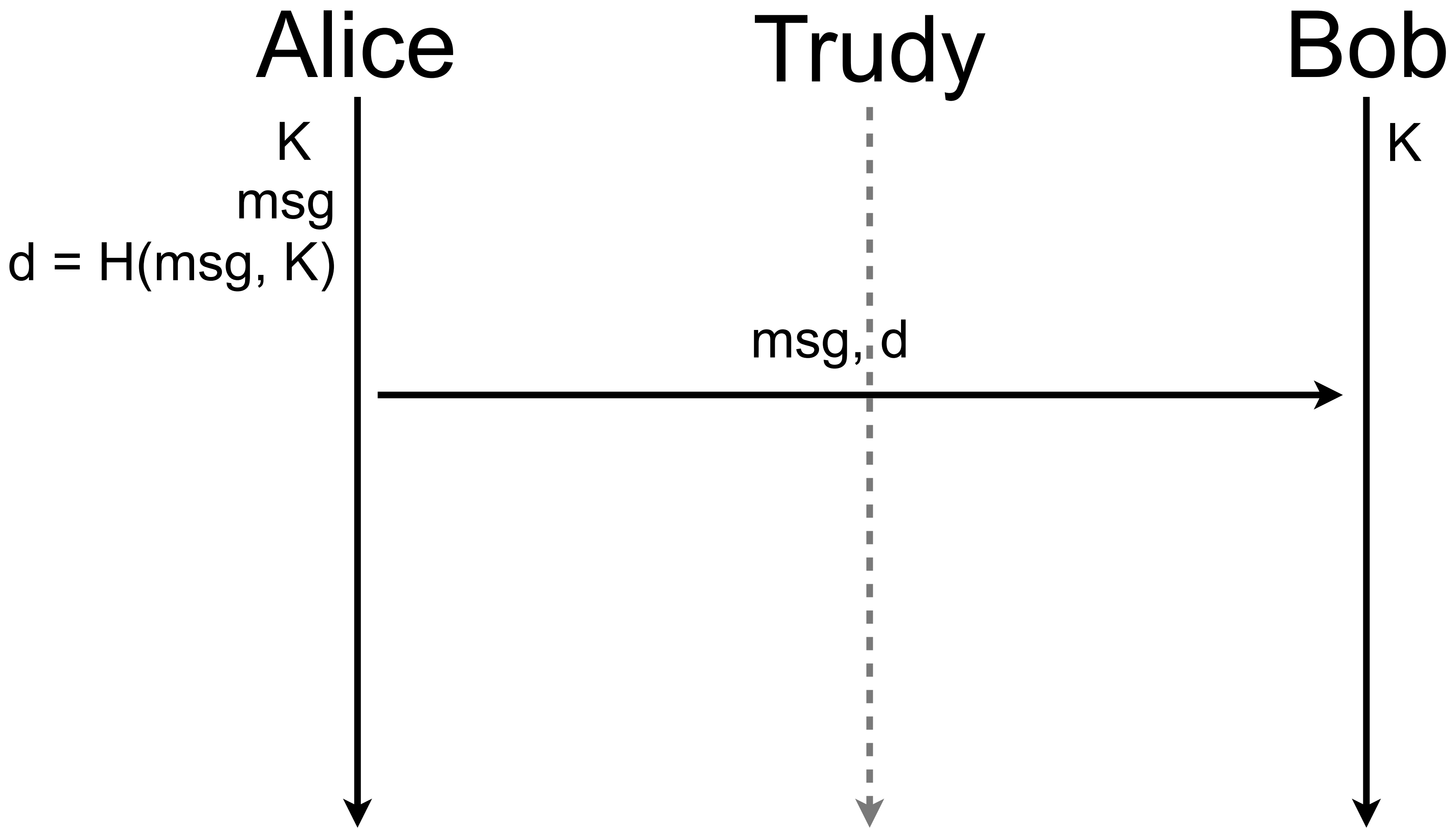
Trudy

Bob

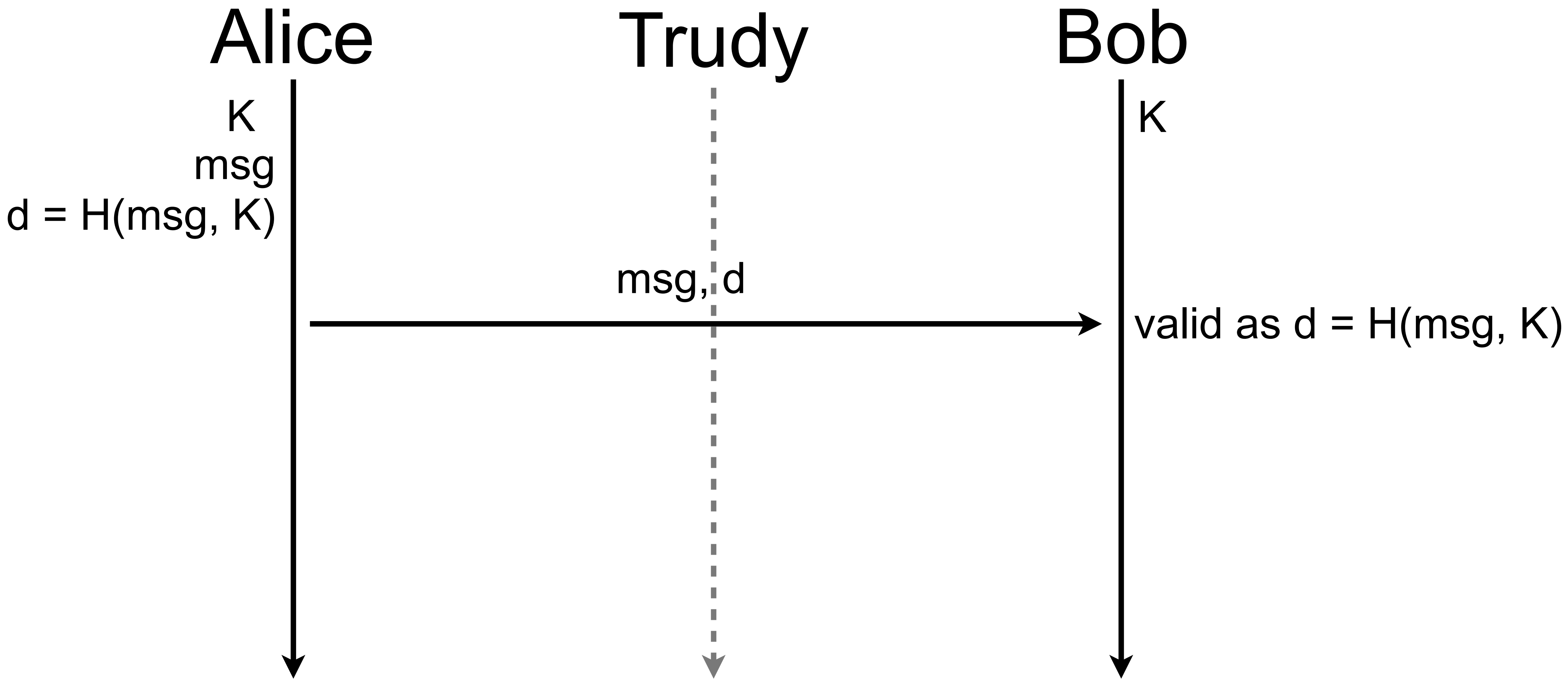
K
msg
d = H(msg, K)

K

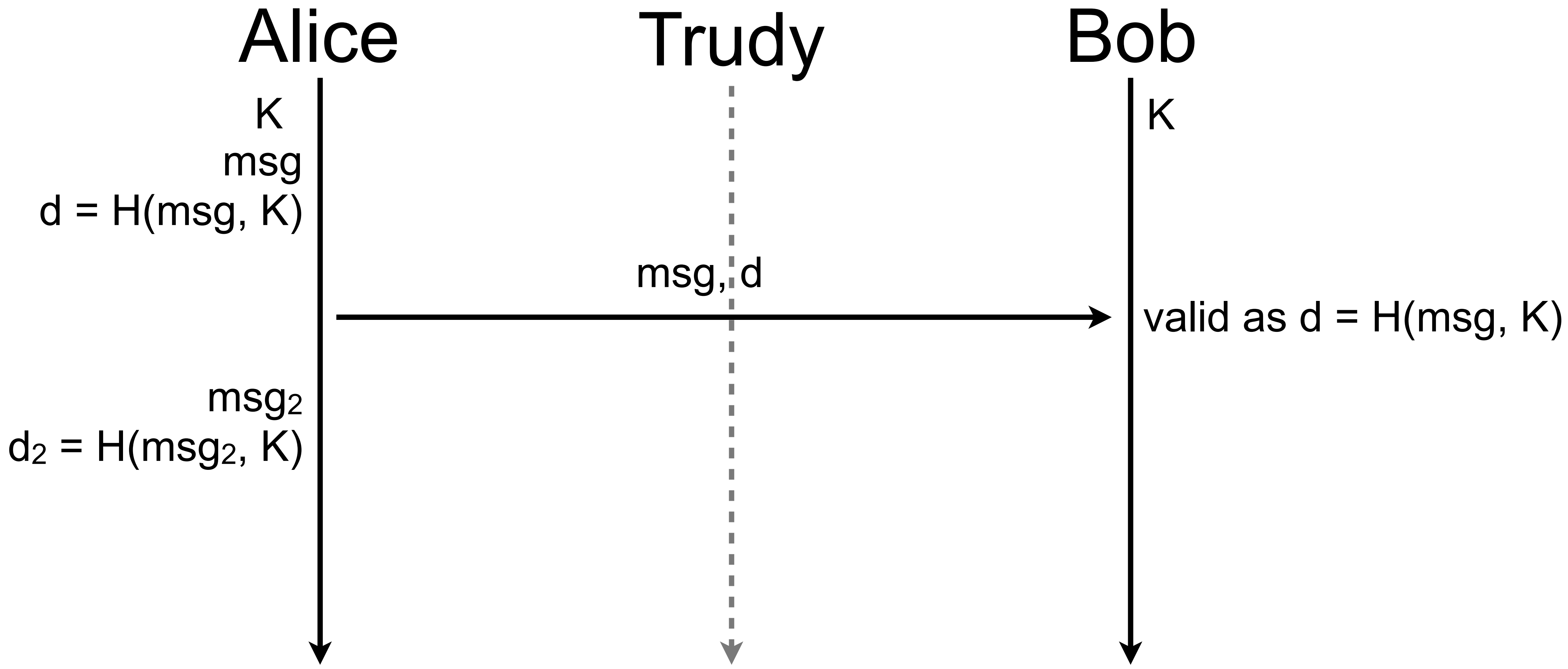
Hash function with salt (contd.)



Hash function with salt (contd.)



Hash function with salt (contd.)



Alice

Trudy

Bob

K

msg

d = H(msg, K)

K

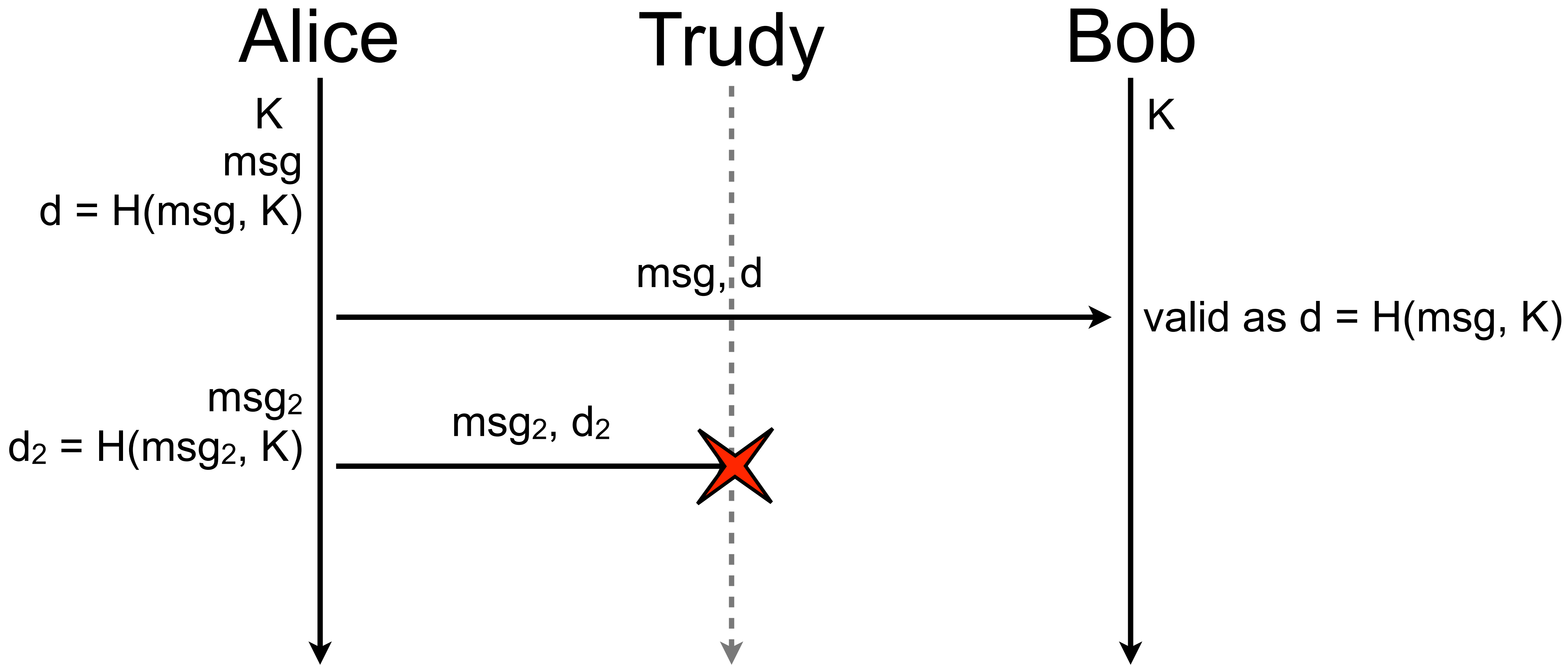
msg, d

valid as d = H(msg, K)

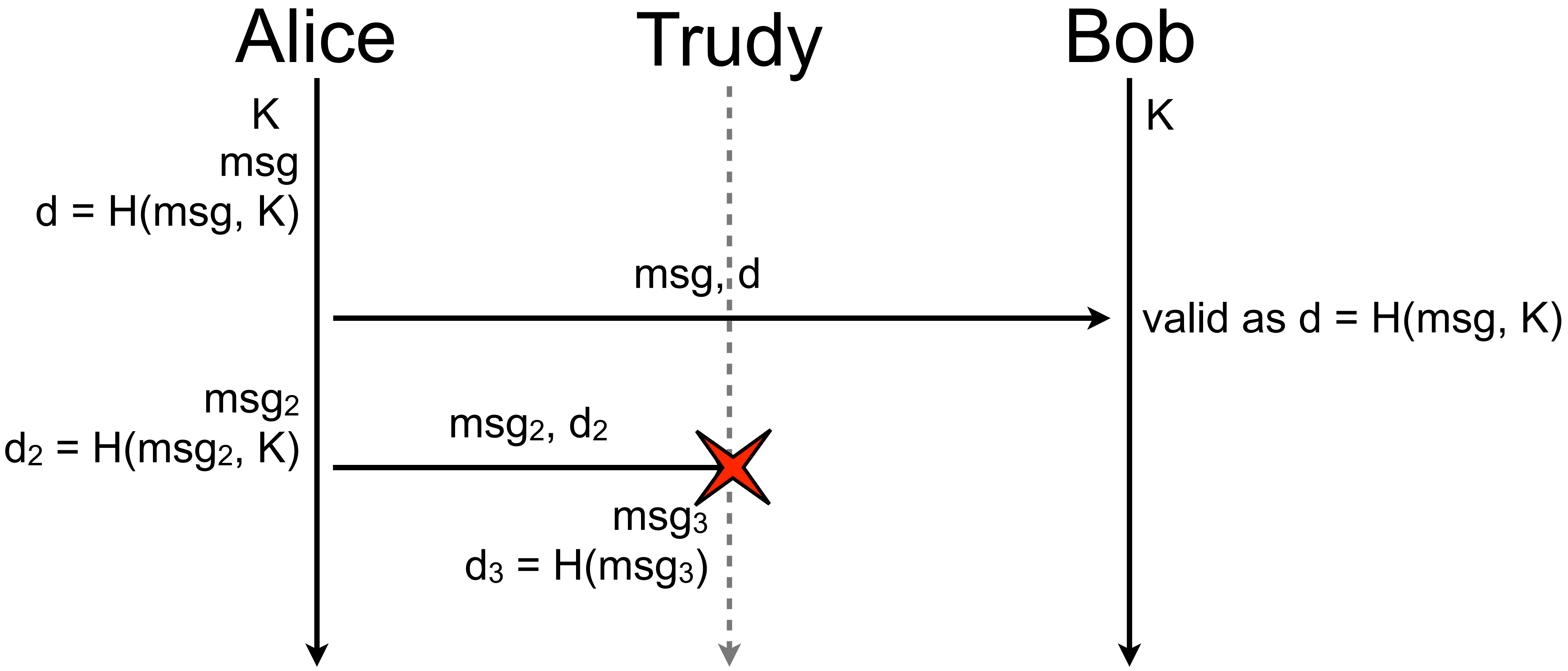
msg_2

$d_2 = H(msg_2, K)$

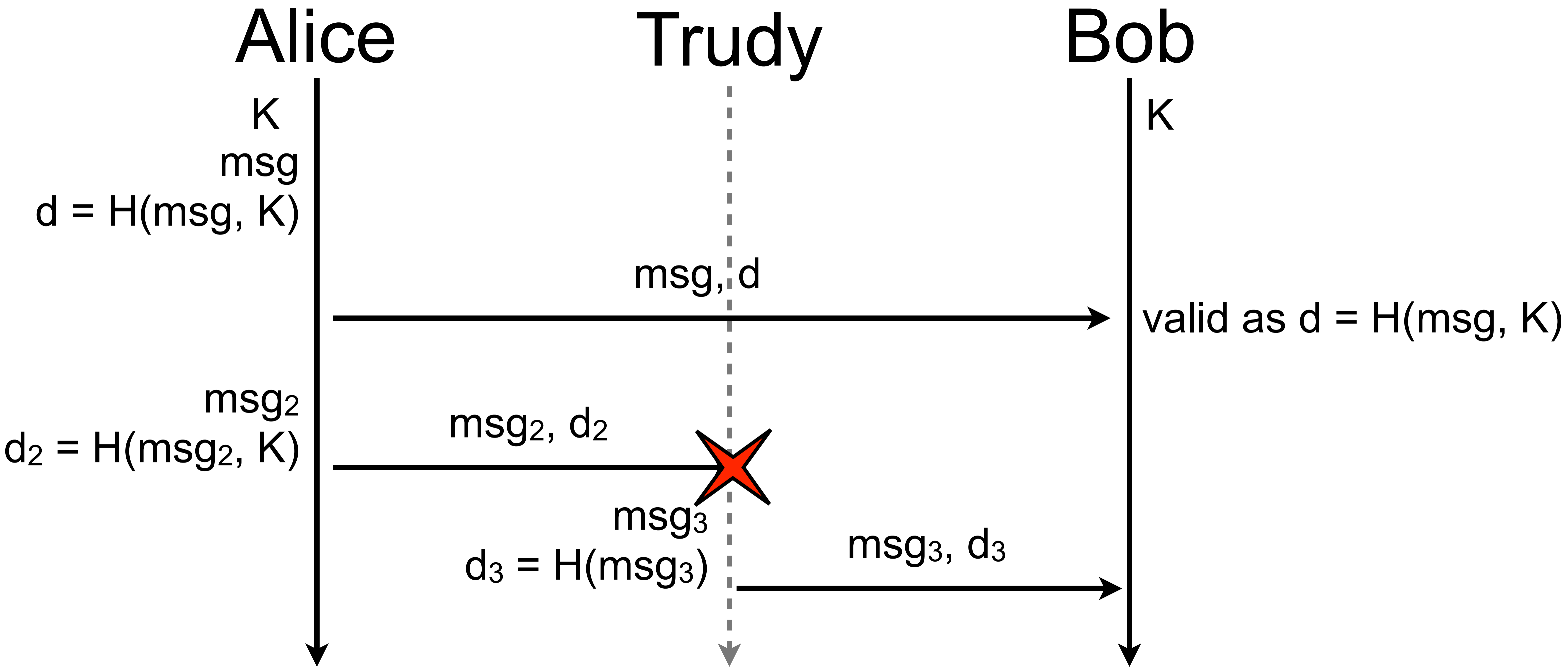
Hash function with salt (contd.)



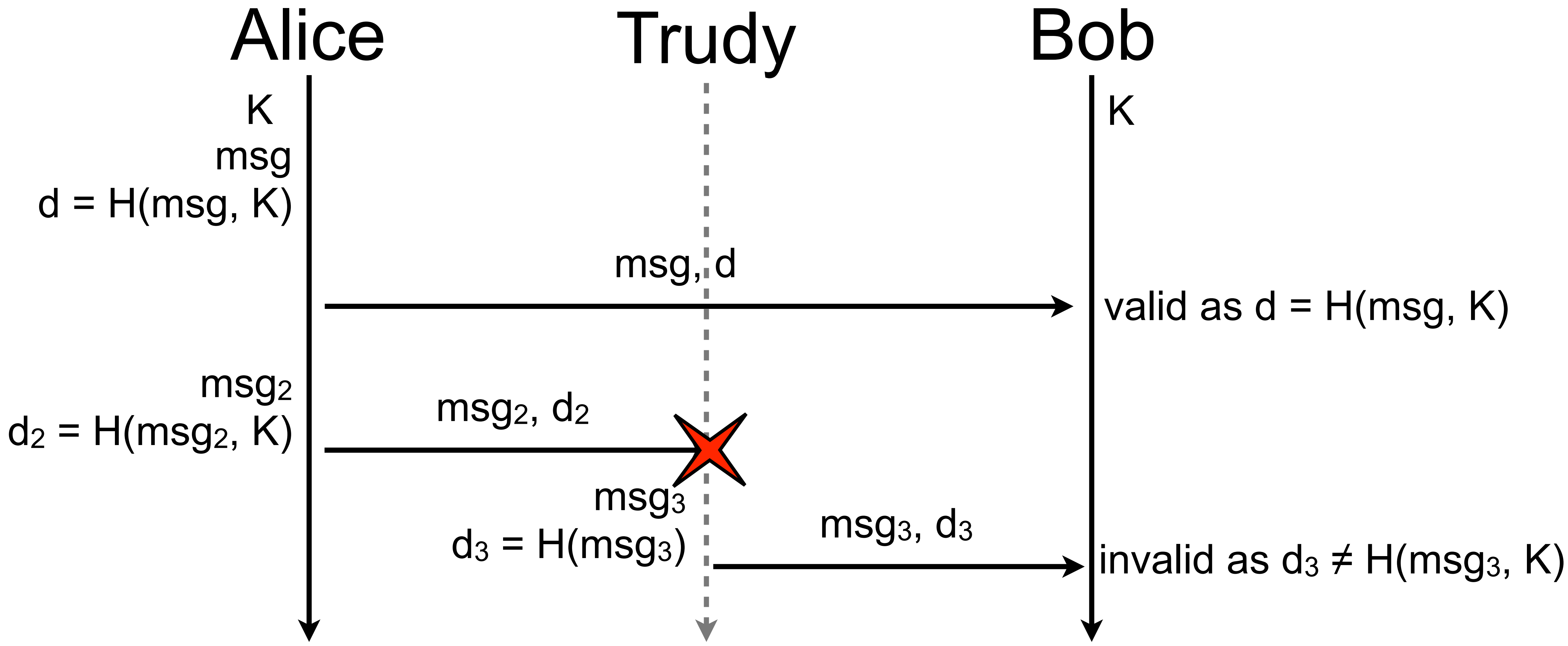
Hash function with salt (contd.)



Hash function with salt (contd.)



Hash function with salt (contd.)



Alice

Trudy

Bob

K

msg

$d = H(msg, K)$

K

msg, d

valid as $d = H(msg, K)$

msg_2

$d_2 = H(msg_2, K)$

msg_2, d_2

msg_3

$d_3 = H(msg_3)$

msg_3, d_3

invalid as $d_3 \neq H(msg_3, K)$

Problem solved?

- fill me
- fill me
- fill me

Problem solved?

- fill me
- fill me
- fill me

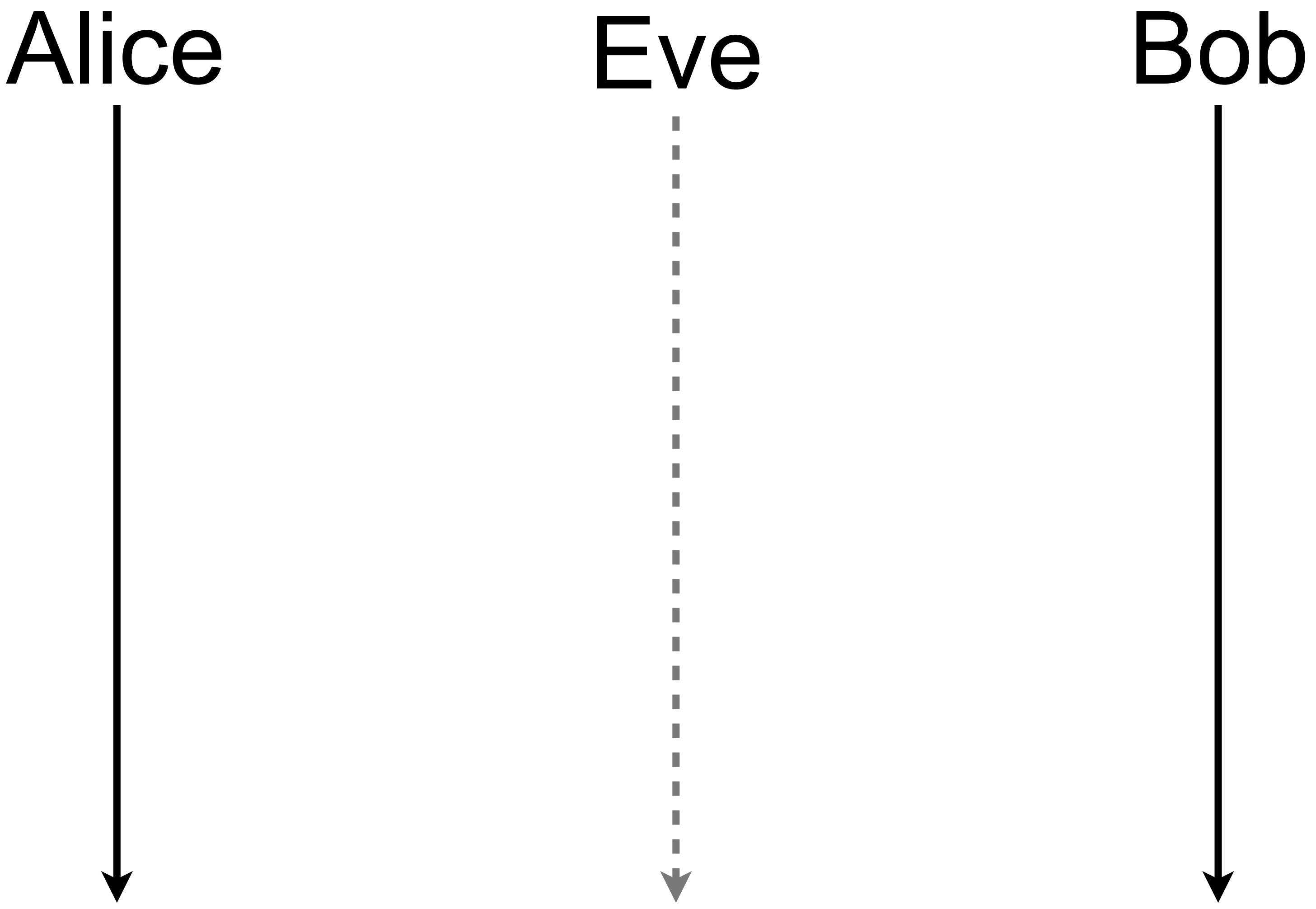
How can Alice and Bob agree on K?

Diffie-Hellman key exchange

- How can Alice and Bob agree on a secret number and be sure that Eve will not discover it?
- Principle
  - do not exchange the secret number but other numbers that are use to build up the secret

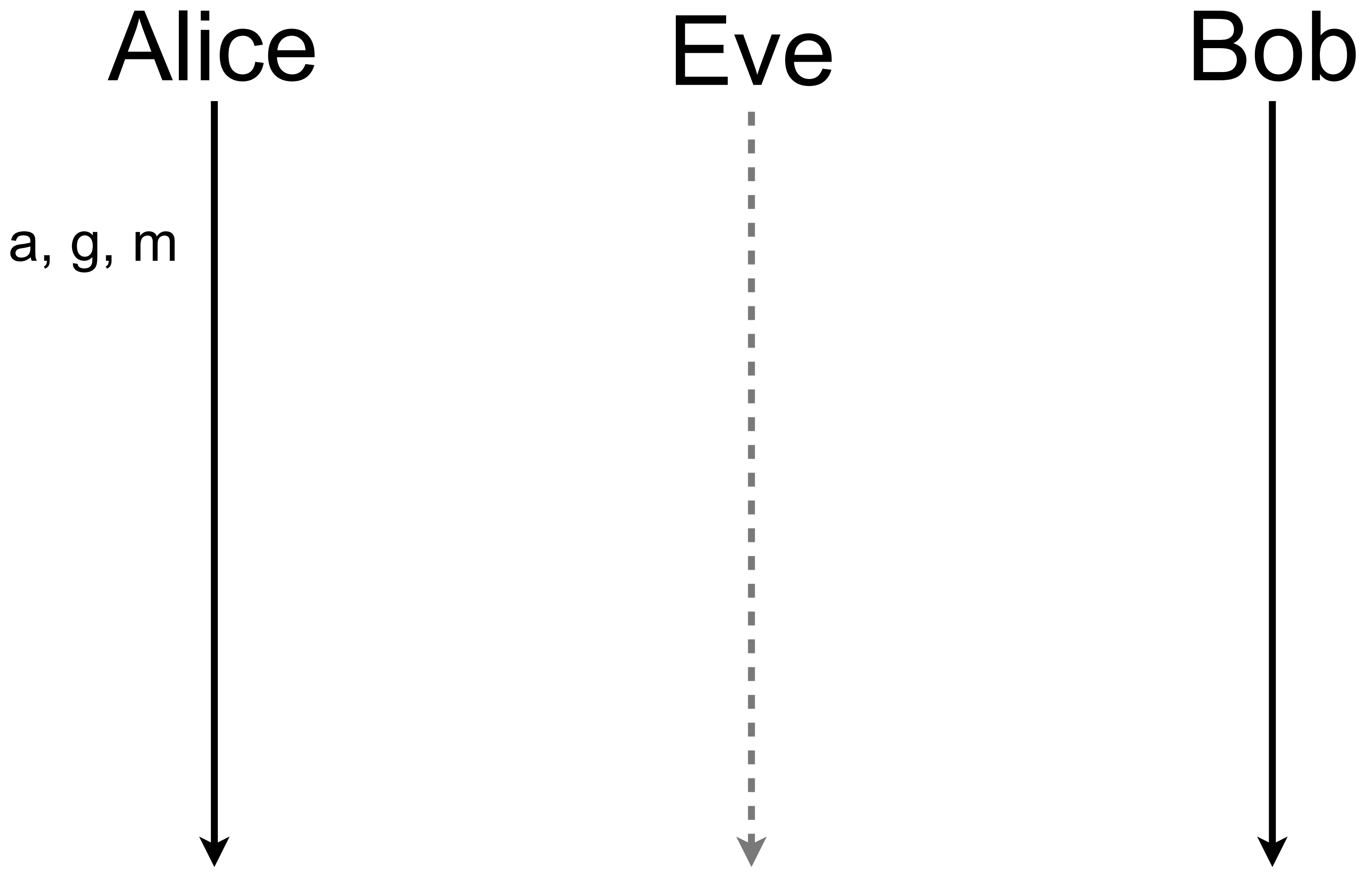
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



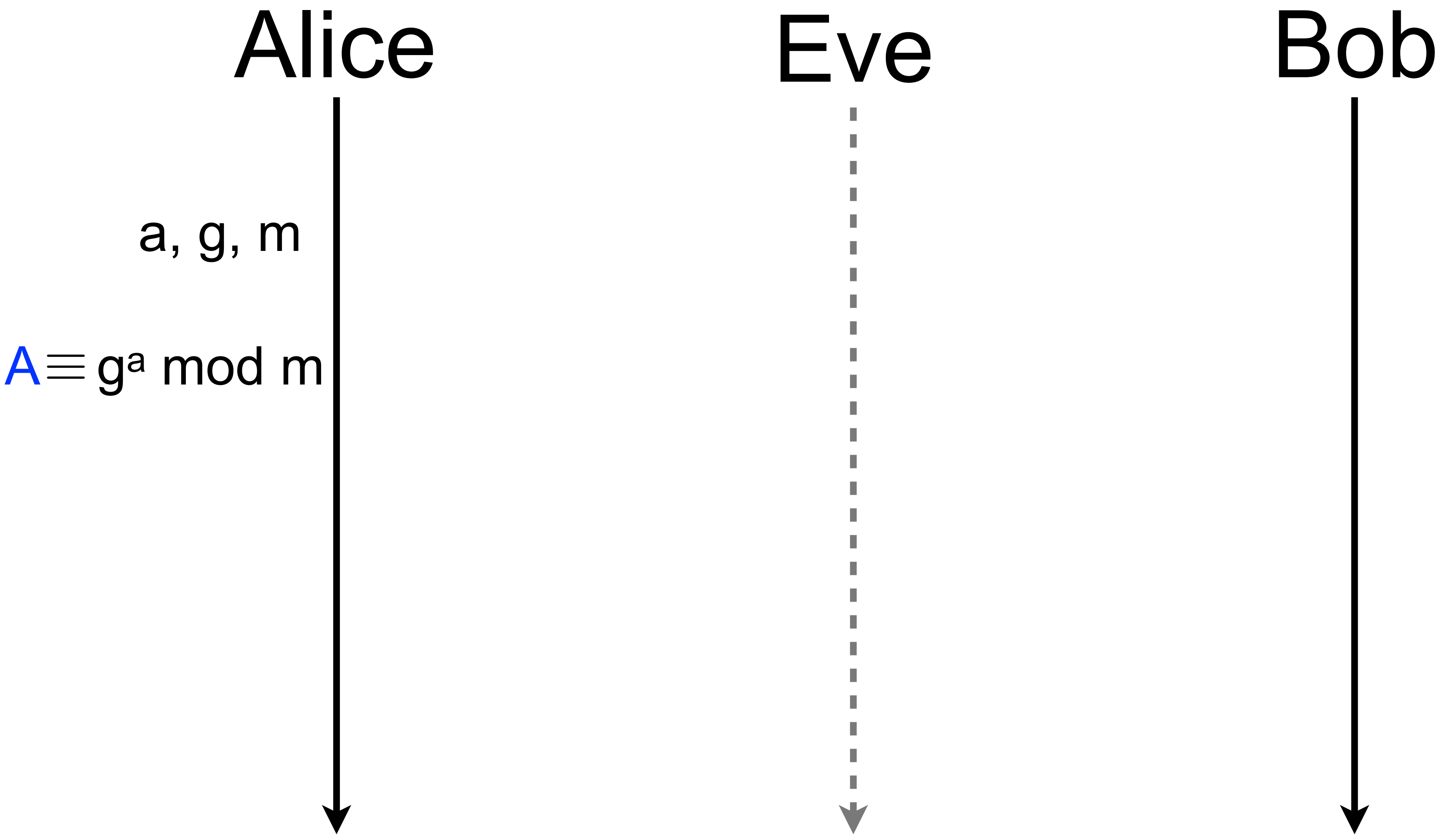
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



Alice

Eve

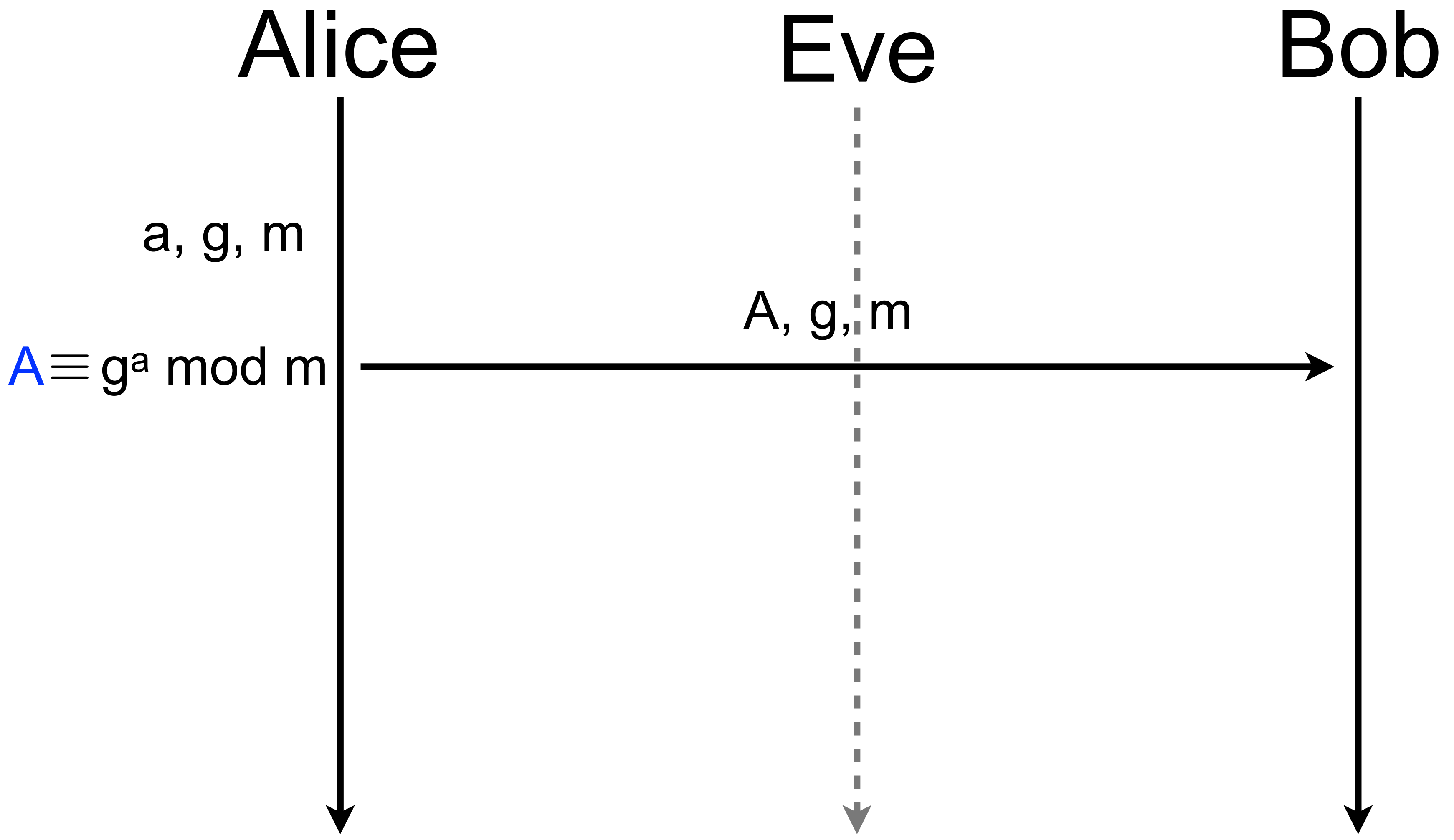
Bob

a, g, m

$A \equiv g^a \bmod m$

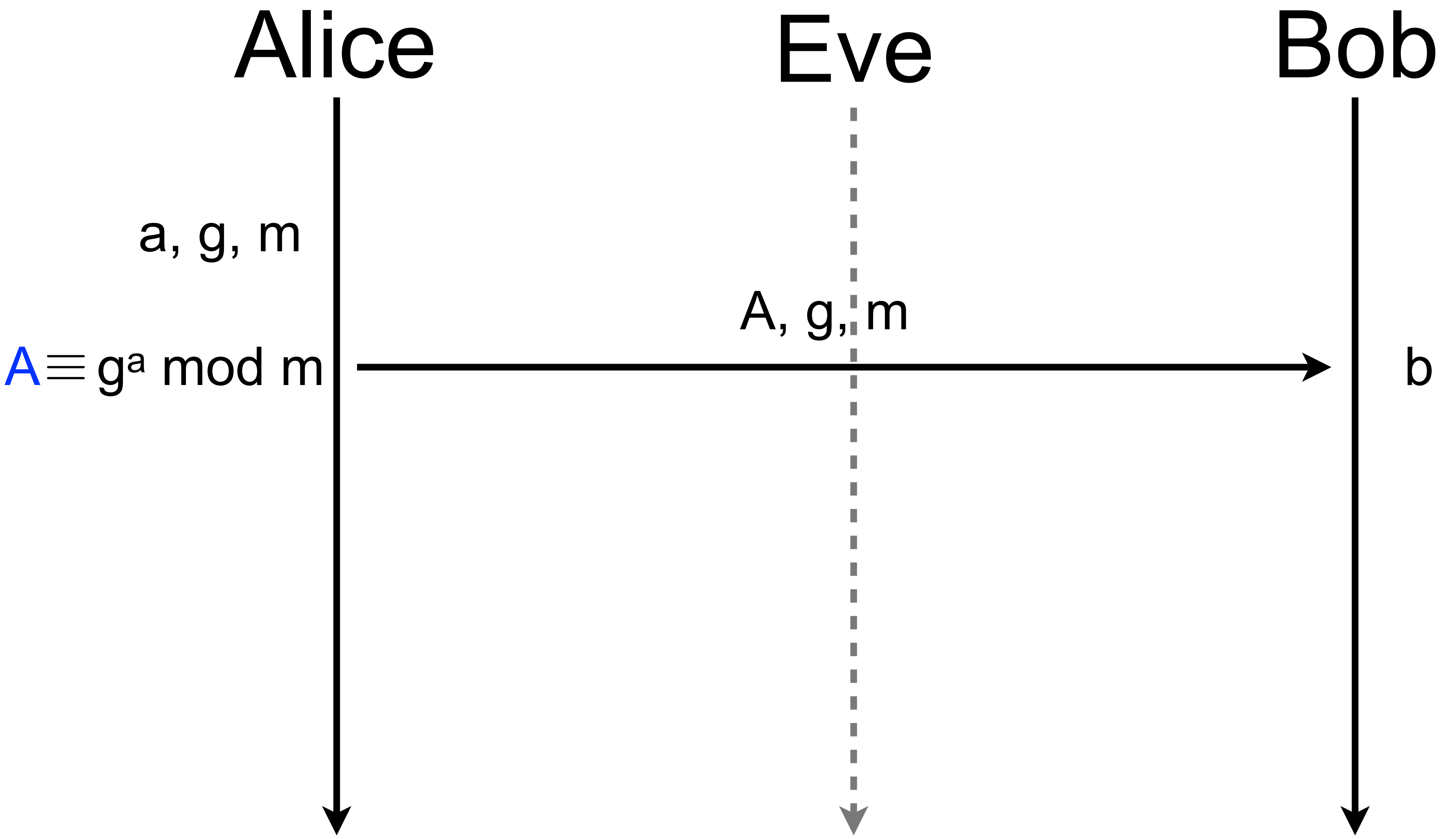
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



Alice

Eve

Bob

a, g, m

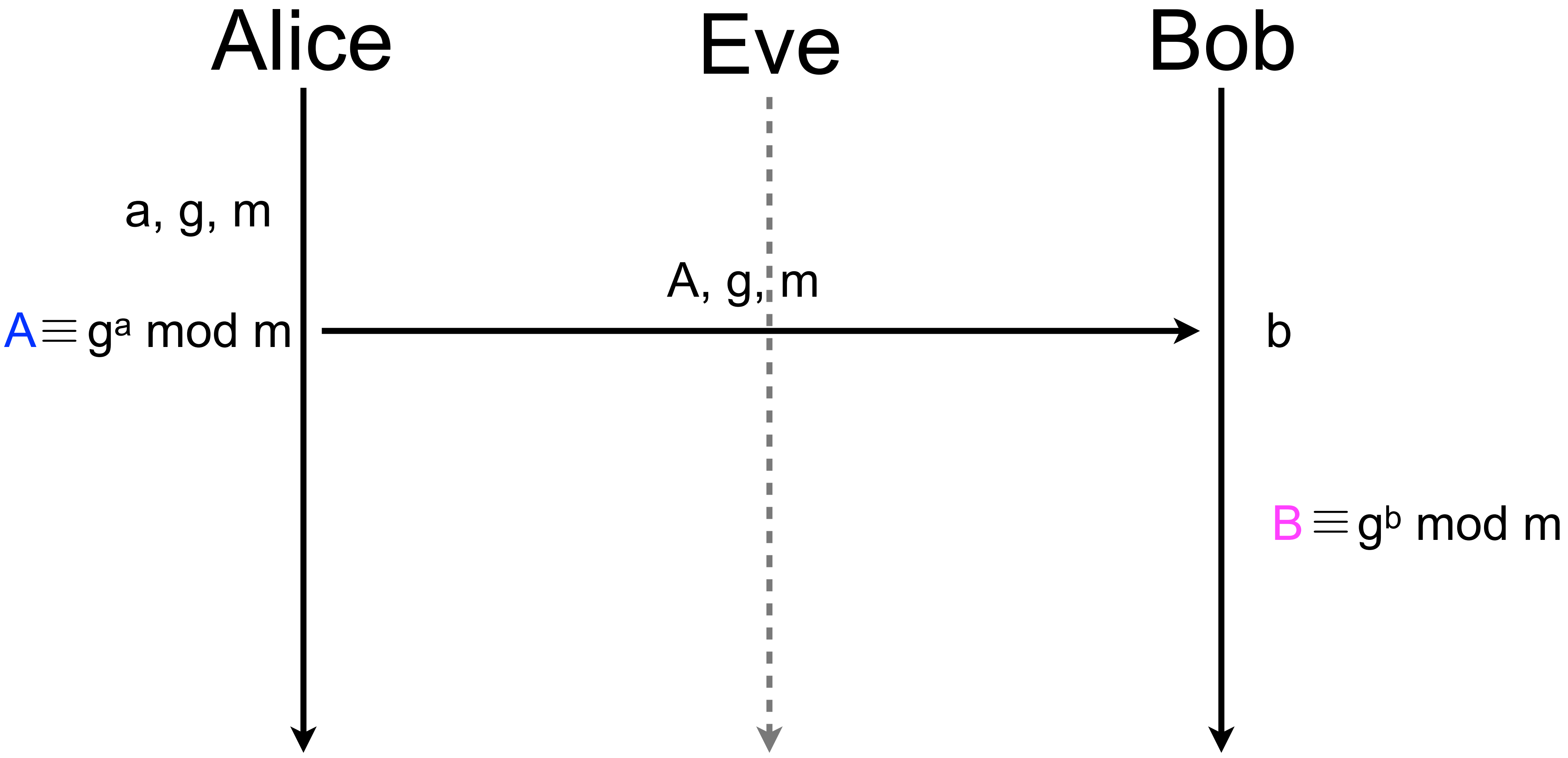
A, g, m

$A \equiv g^a \bmod m$

b

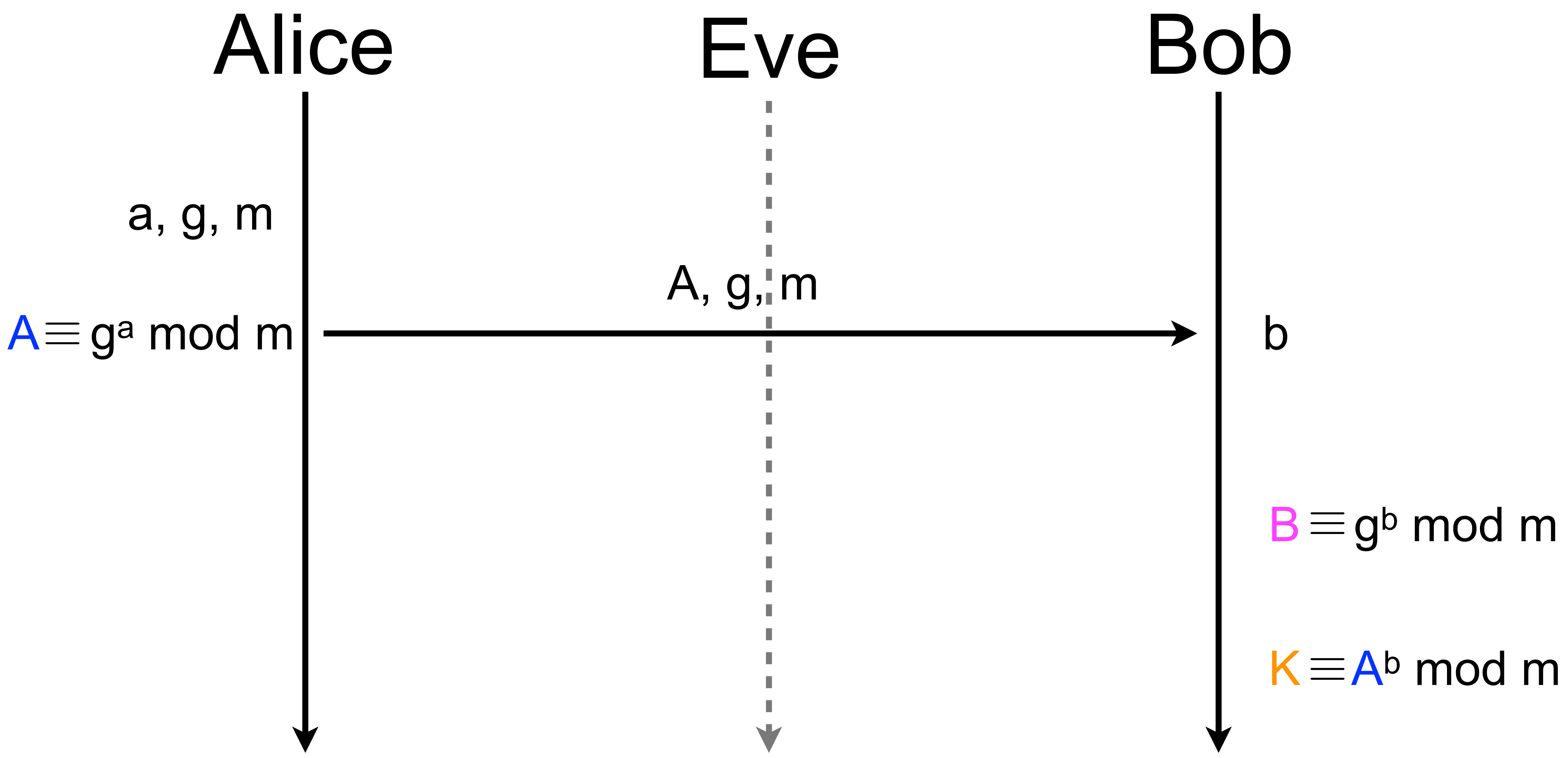
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



Alice

Eve

Bob

a, g, m

$A \equiv g^a \bmod m$

A, g, m

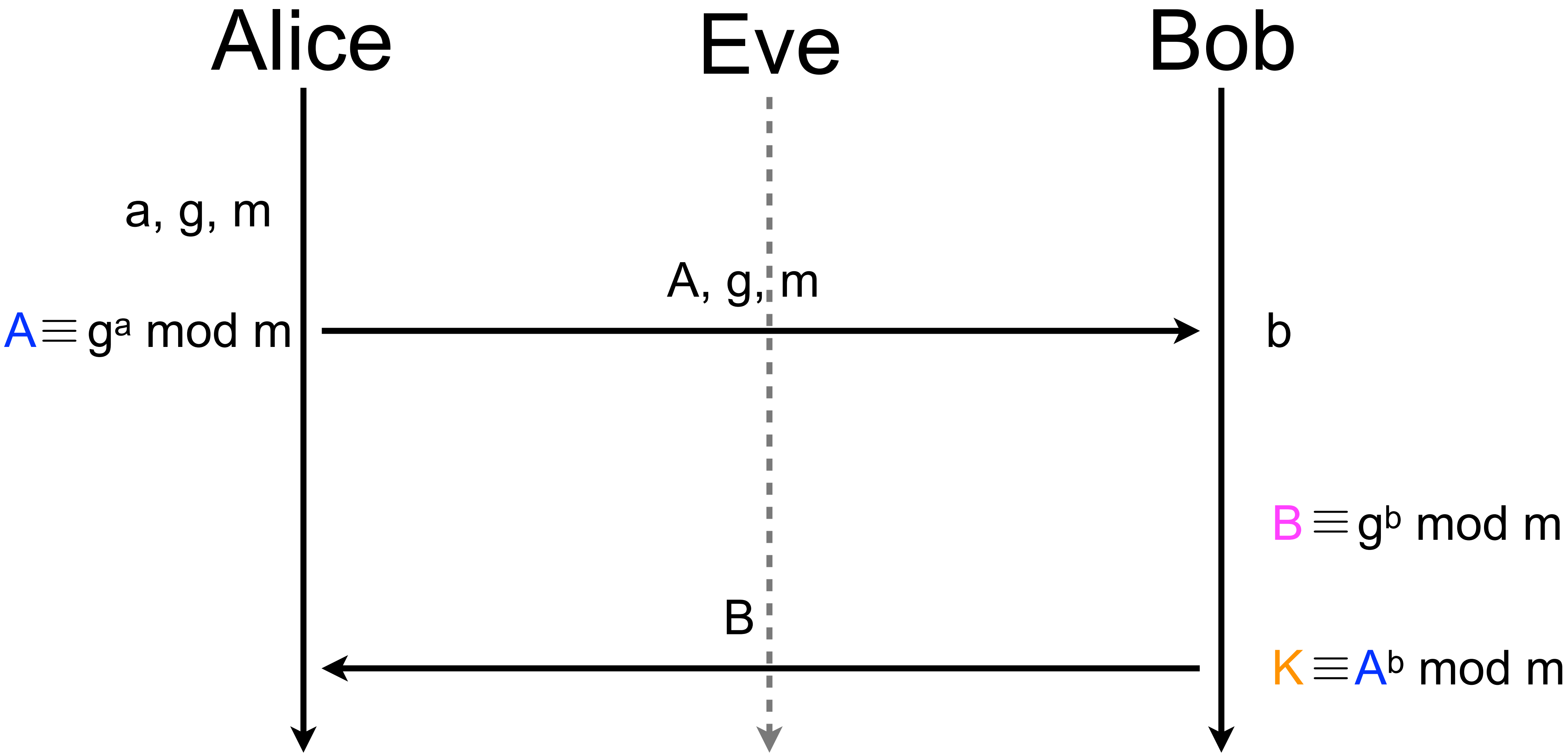
b

$B \equiv g^b \bmod m$

$K \equiv A^b \bmod m$

Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



Alice

Eve

Bob

a, g, m

$A \equiv g^a \bmod m$

A, g, m

b

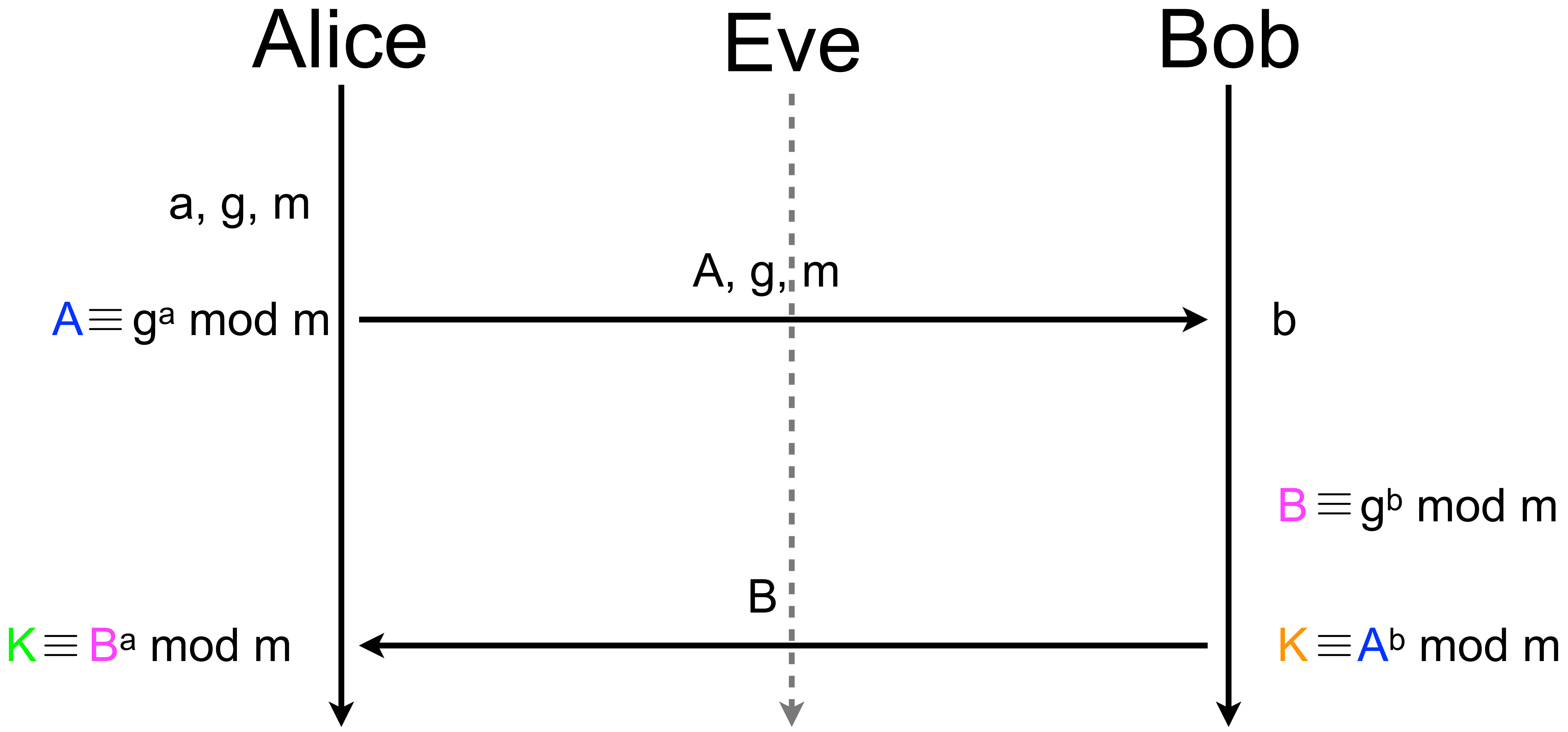
$B \equiv g^b \bmod m$

B

$K \equiv A^b \bmod m$

Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



Alice

Eve

Bob

a, g, m

A, g, m

$A \equiv g^a \bmod m$

b

$B \equiv g^b \bmod m$

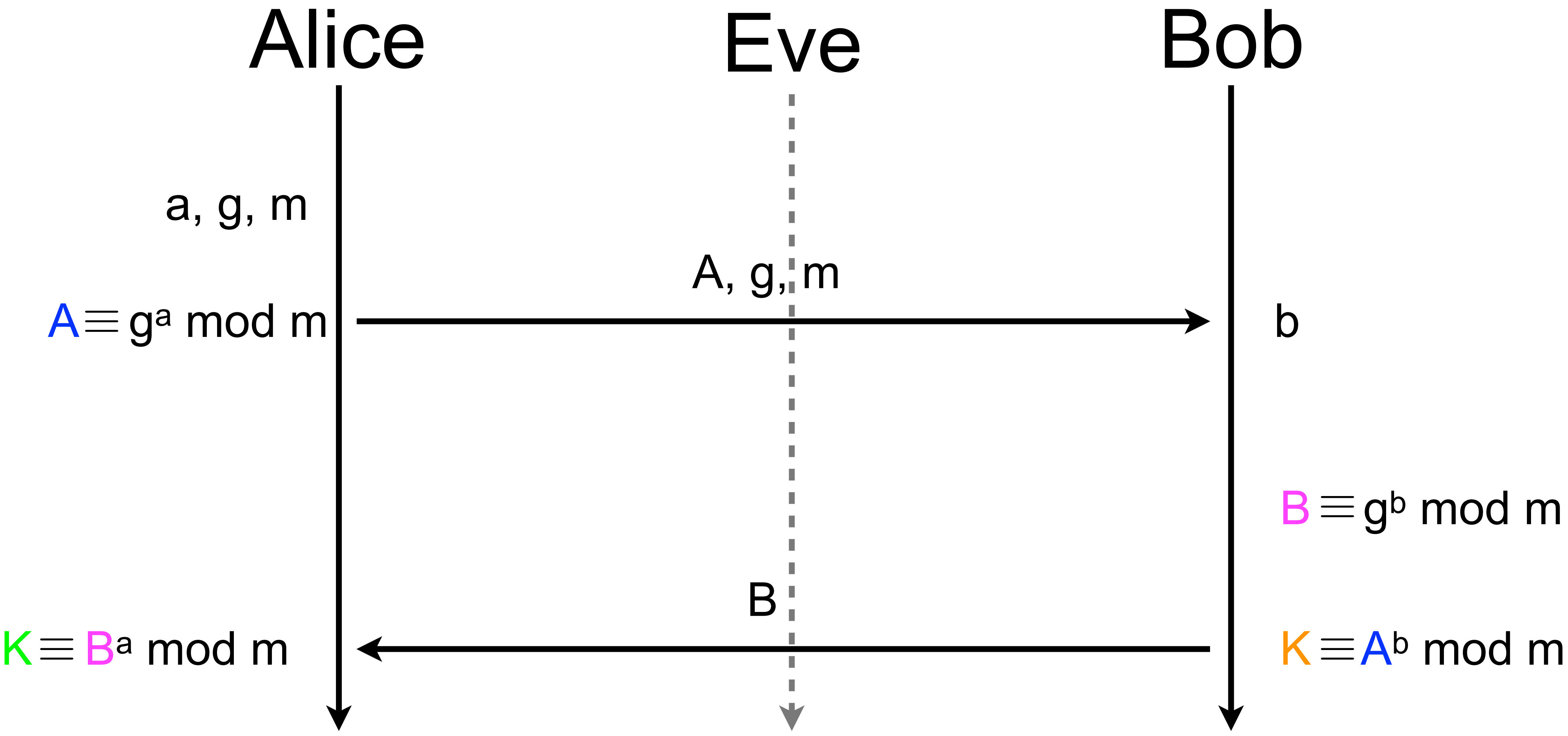
B

$K \equiv B^a \bmod m$

$K \equiv A^b \bmod m$

Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



Alice

Eve

Bob

a, g, m

$A \equiv g^a \bmod m$

A, g, m

b

$B \equiv g^b \bmod m$

B

$K \equiv B^a \bmod m$

$K \equiv A^b \bmod m$

$K \equiv A^b \bmod m \equiv (g^a \bmod m)^b \bmod m \equiv g^{ba} \bmod m \equiv (g^b \bmod m)^a \bmod m \equiv B^a \bmod m \equiv K$

Diffie-Hellman key exchange (contd.)

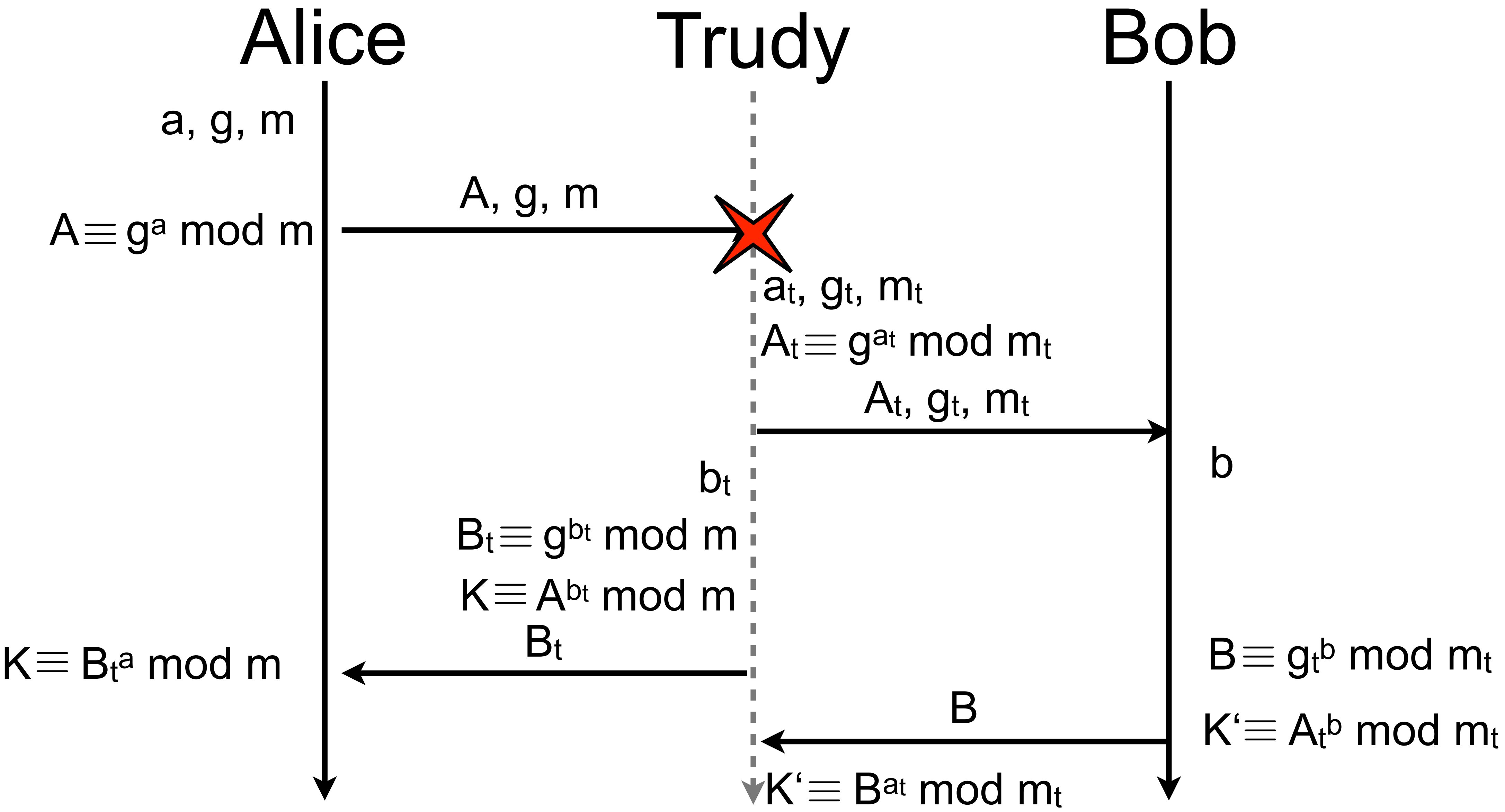
- Why can’t Eve guess K if she knows A, B, g, and m?
  - discrete exponentiation is linear with the size of the argument
    - easy to compute $x \equiv y^{z} \bmod p$
  - but for some discrete groups, no efficient algorithm is known to compute discrete logarithm
    - hard to determine natural z that ensures $x \equiv y^{z} \bmod p$
  - Eve knows A, B, g, and m but can’t determine neither **a** nor **b** that are absolutely necessary to compute K
    - $K \equiv A^{\mathbf{b}} \bmod m \equiv (g^{\mathbf{a}} \bmod m)^{\mathbf{b}} \bmod p \equiv g^{\mathbf{ba}} \bmod m$
      $\equiv (g^{\mathbf{b}} \bmod m)^{\mathbf{a}} \bmod m \equiv B^{\mathbf{a}} \bmod m$

3-Diffie Hellman

- Everyone
  - $A = g^a \bmod m$
  - $B = g^b \bmod m$
  - $C = g^c \bmod m$
- On A
  - $AC = C^a \bmod m$, $AB = B^a \bmod m$
- B
  - $BC = C^b \bmod m$
- On A
  - $K = BC^a \bmod m$
- On B
  - $K = AC^b \bmod m$
- On C
  - $K = AB^c \bmod m$
- “Proof”
  - $K = AB^c \bmod m = B^{a\ c} \bmod m = g^{b\ ac} \bmod m = B^{abc} \bmod m$

Diffie-Hellman key exchange (contd.)

- Trudy can break Diffie-Hellman



Alice

Trudy

Bob

a, g, m

A, g, m

$A \equiv g^a \bmod m$

a_t, g_t, m_t

$A_t \equiv g^{a_t} \bmod m_t$

A_t, g_t, m_t

b

b_t

$B_t \equiv g^{b_t} \bmod m$

$K \equiv A^{b_t} \bmod m$

B_t

$K \equiv B_t^a \bmod m$

$B \equiv g_t^b \bmod m_t$

B

$K' \equiv A_t^b \bmod m_t$

$K' \equiv B^{a_t} \bmod m_t$

Diffie-Hellman key exchange (contd.)

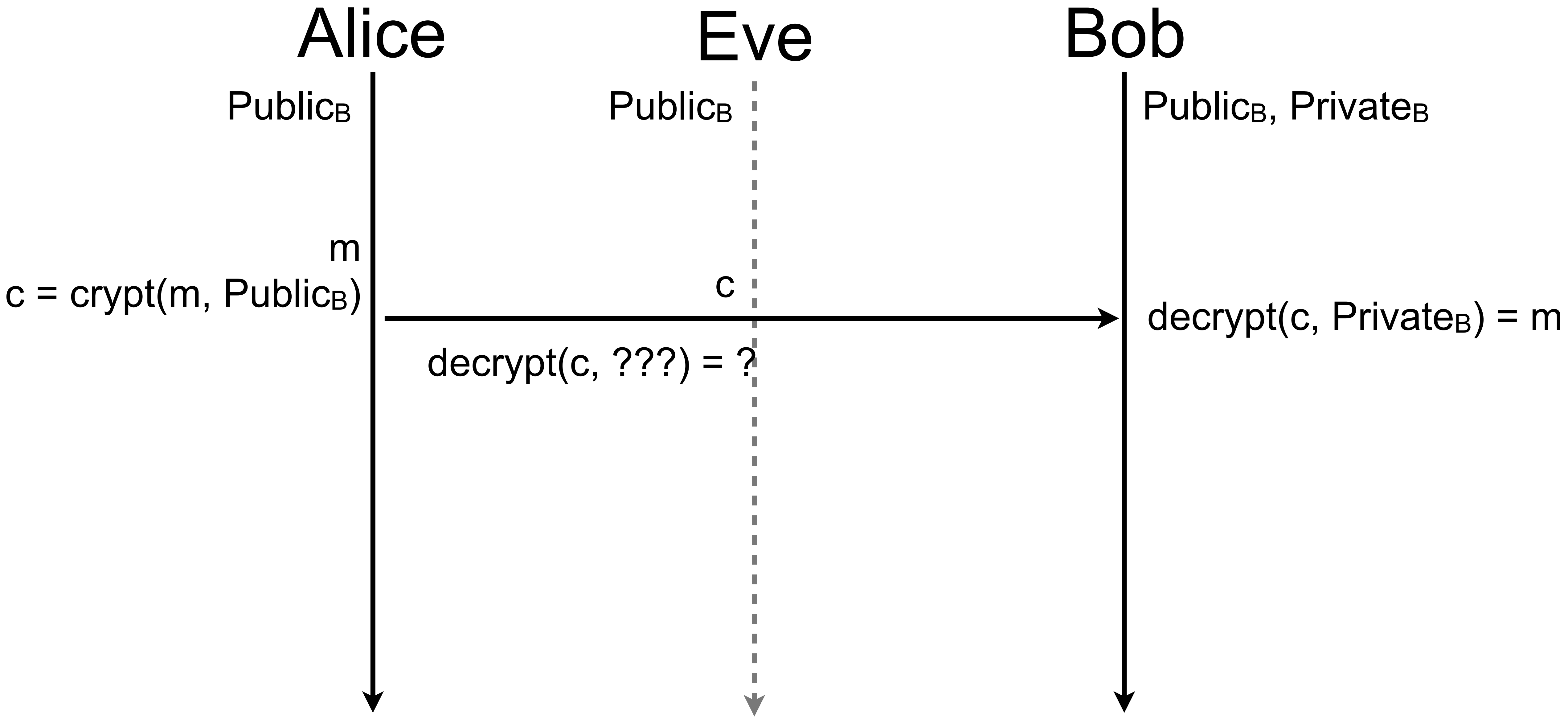
- How can we protect Diffie-Hellman from Trudy?
- Principle
  - Alice and Bob sign the messages exchanged in Diffie-Hellman (?!)

Asymmetric cryptography

- In asymmetric cryptography (aka public-key cryptography), two keys are used
  - public key
    - publicly available to anybody (even attackers)
    - used to encrypt a message
  - private key
    - known only by the legitimate owner of the public key
    - used to decrypt a message
- e.g., RSA, PGP, Diffie-Hellman
- Public-key cryptography is 10 to 100 times slower than symmetric-key cryptography
  - seldom (never?) used to encrypt communications

Asymmetric cryptography (contd.)

- Eve cannot determine the message



Alice

Eve

Bob

Public_B

Public_B

Public_B, Private_B

m

c = crypt(m, Public_B)

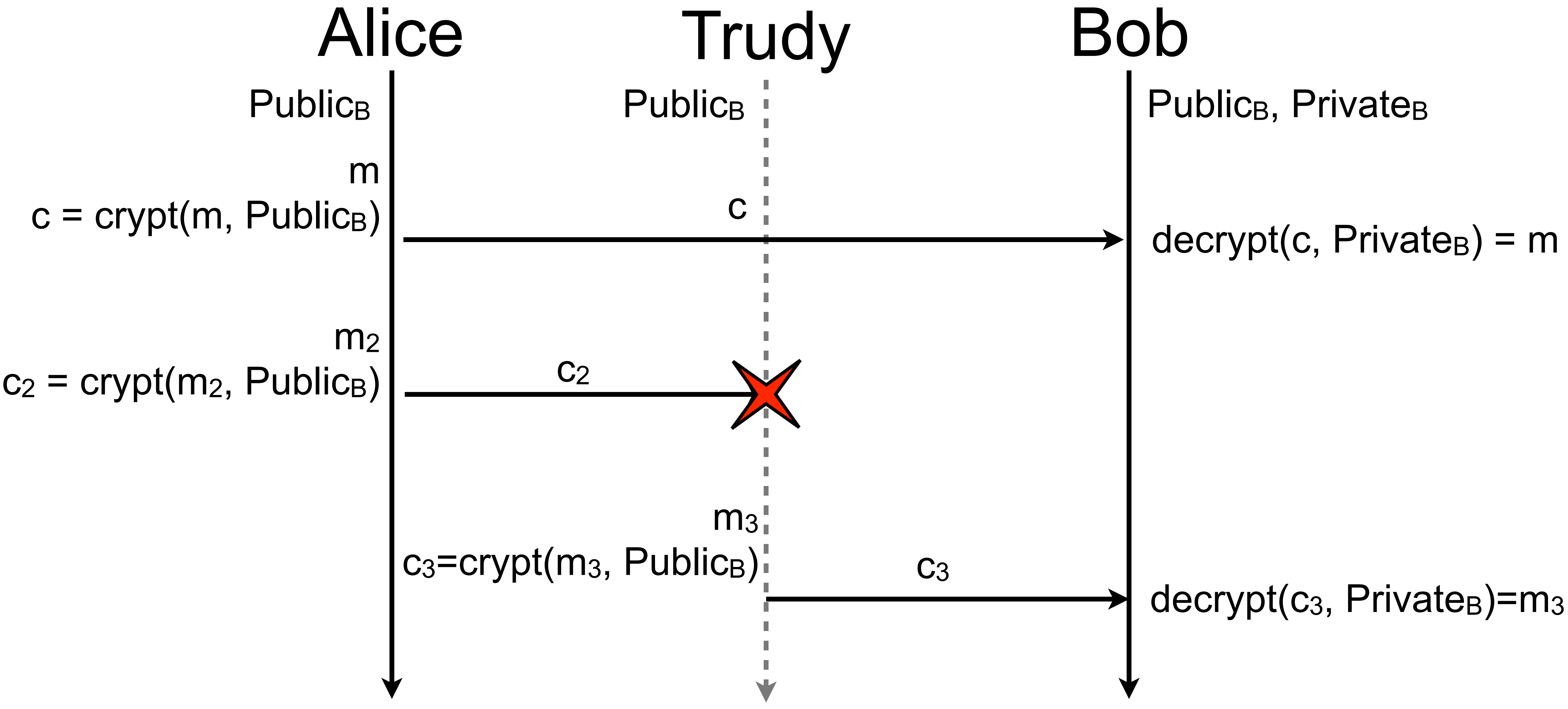
c

decrypt(c, Private_B) = m

decrypt(c, ???) = ?

Asymmetric cryptography (contd.)

- Trudy can send a forged message



Alice

Trudy

Bob

$Public_B$

$Public_B$

$Public_B$, $Private_B$

m

c = crypt(m, $Public_B$)

c

decrypt(c, $Private_B$) = m

m_2

c_2 = crypt(m_2, $Public_B$)

c_2

m_3

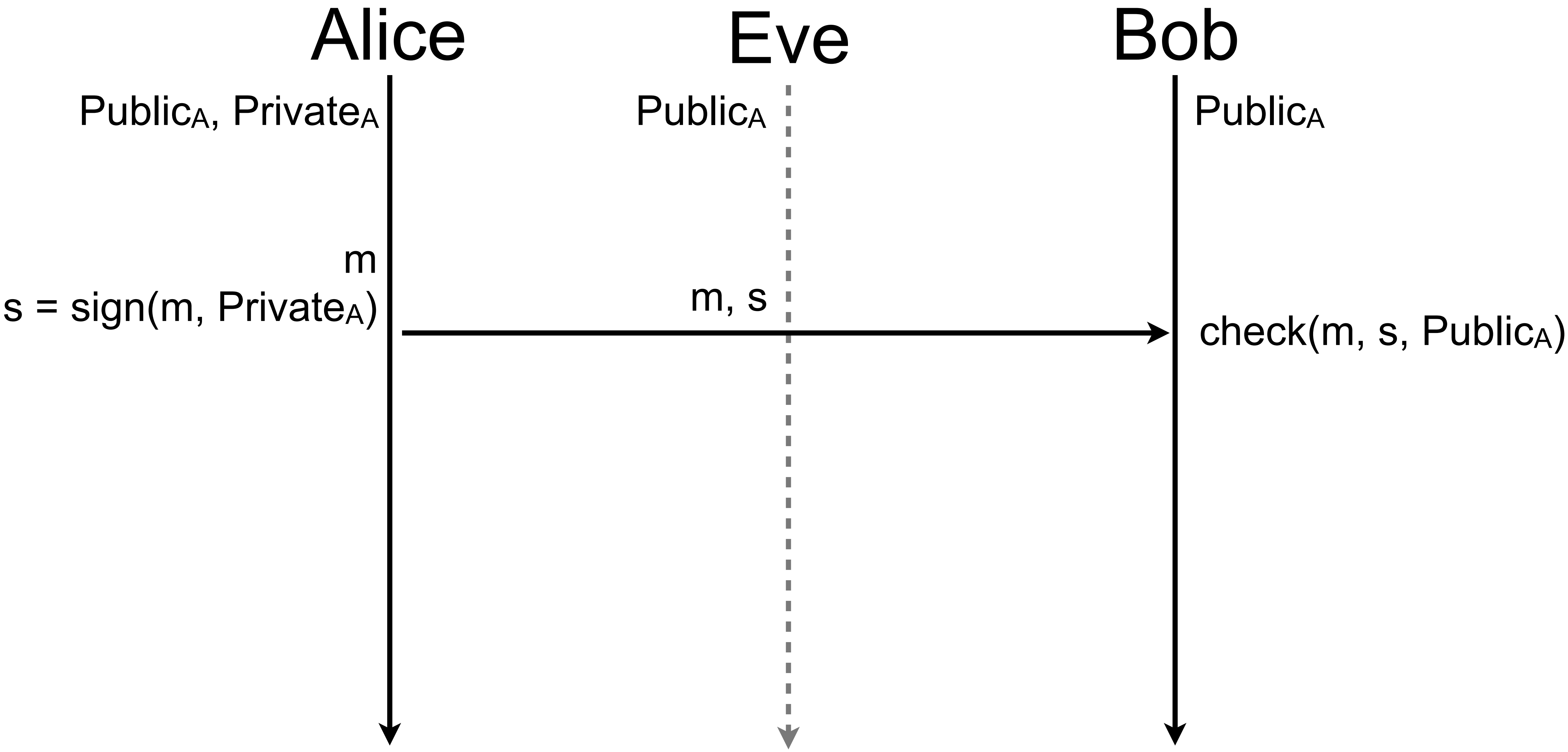
c_3=crypt(m_3, $Public_B$)

c_3

decrypt(c_3, $Private_B$)=m_3

Asymmetric cryptography (contd.)

- Eve can read the message



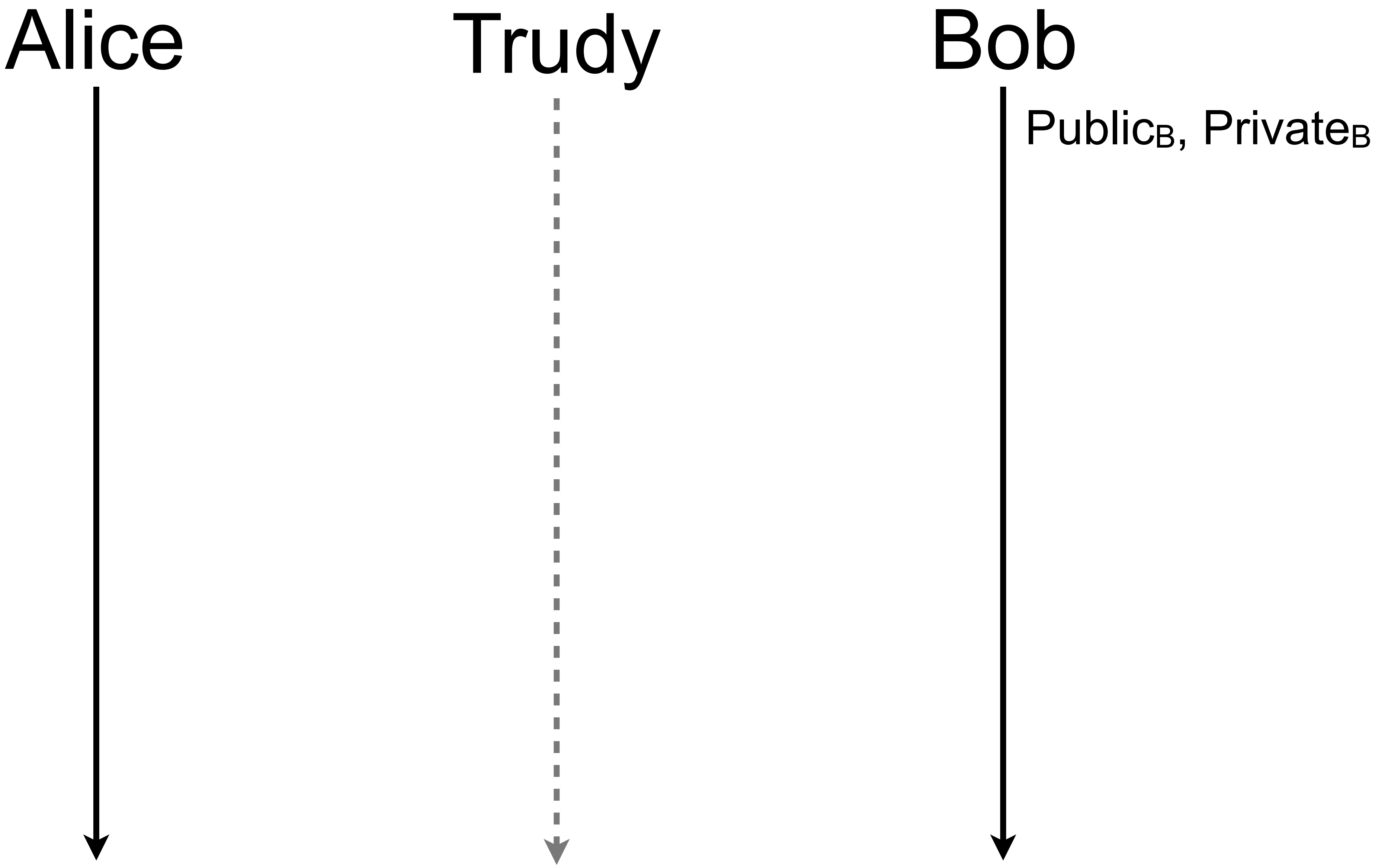
Alice
Eve
Bob
Public_A, Private_A
Public_A
Public_A
m
s = sign(m, Private_A)
m, s
check(m, s, Public_A)

How to build sign and check?

- s = sign(H(m), k) = crypt(H(m), k)
- check(m, s, K) = (H(m)==decrypt(s, K))
  - where k is the private key of the signer and K is the public key
- Asymmetric cryptography is slow and m can be large
  - encrypting m would be too costly
  - solution: consider the digest of m while signing

Public key infrastructure

- How to safely obtain Bob’s public key?



Alice

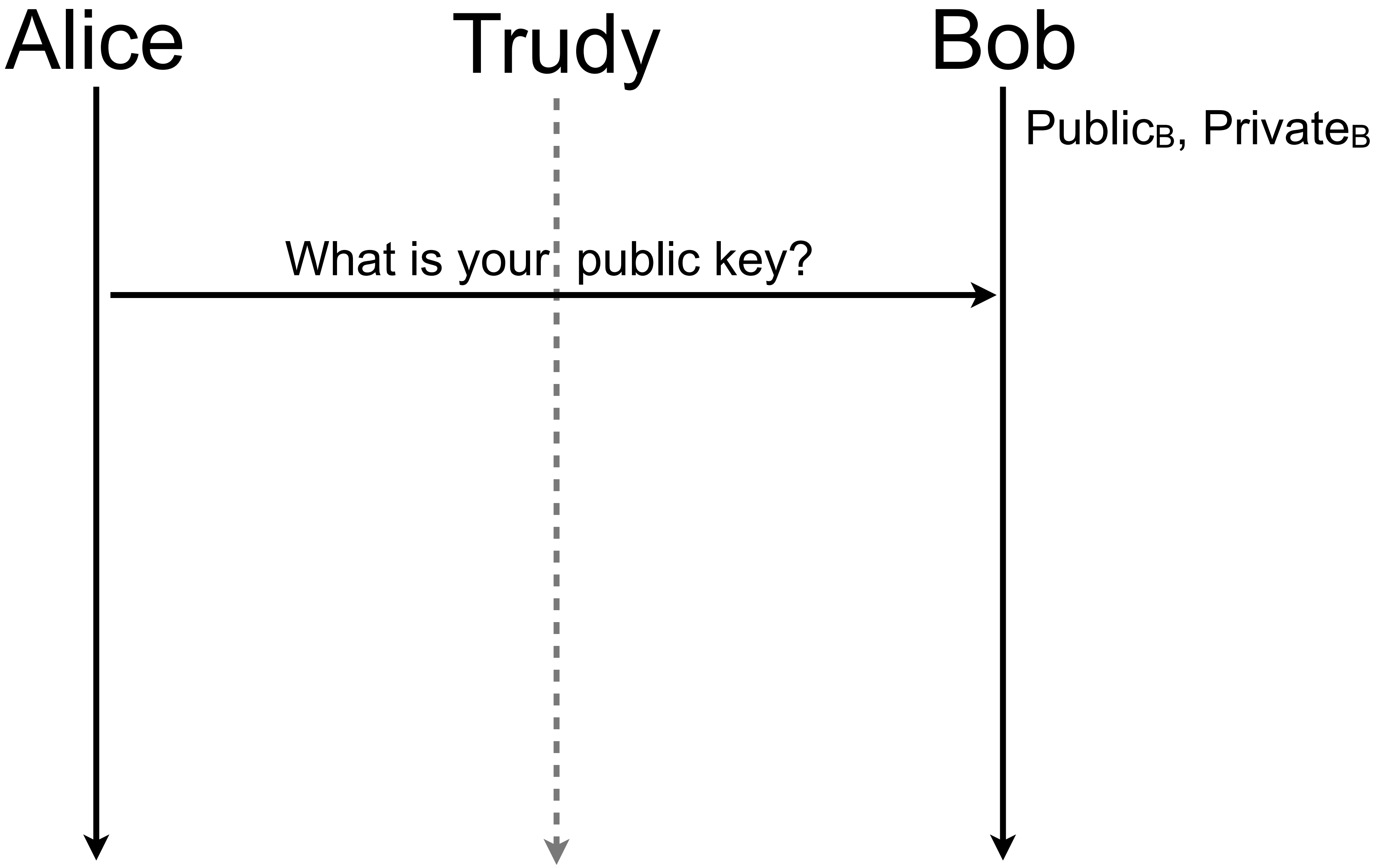
Trudy

Bob

Public_B, Private_B

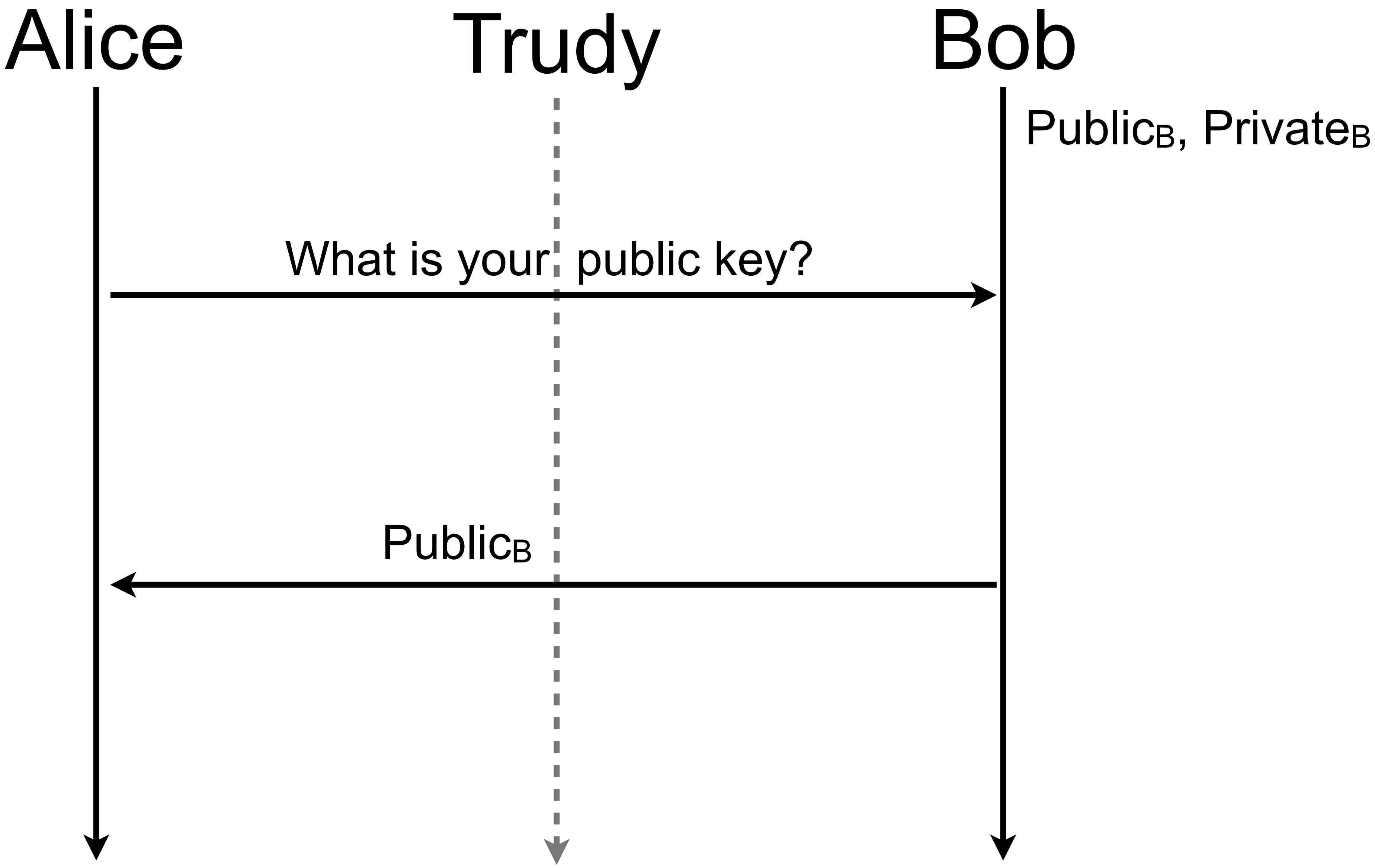
Public key infrastructure

- How to safely obtain Bob's public key?



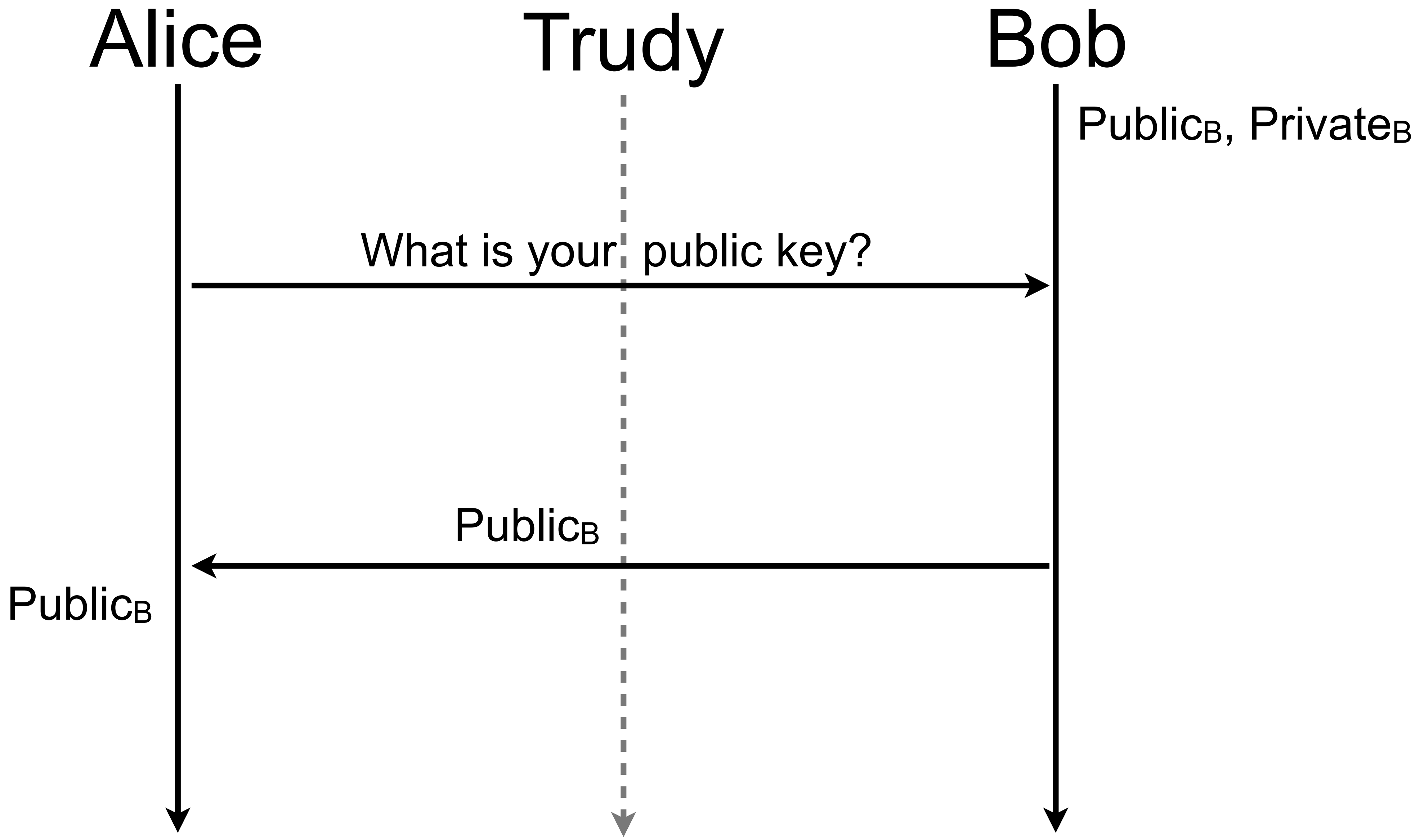
Public key infrastructure

- How to safely obtain Bob’s public key?



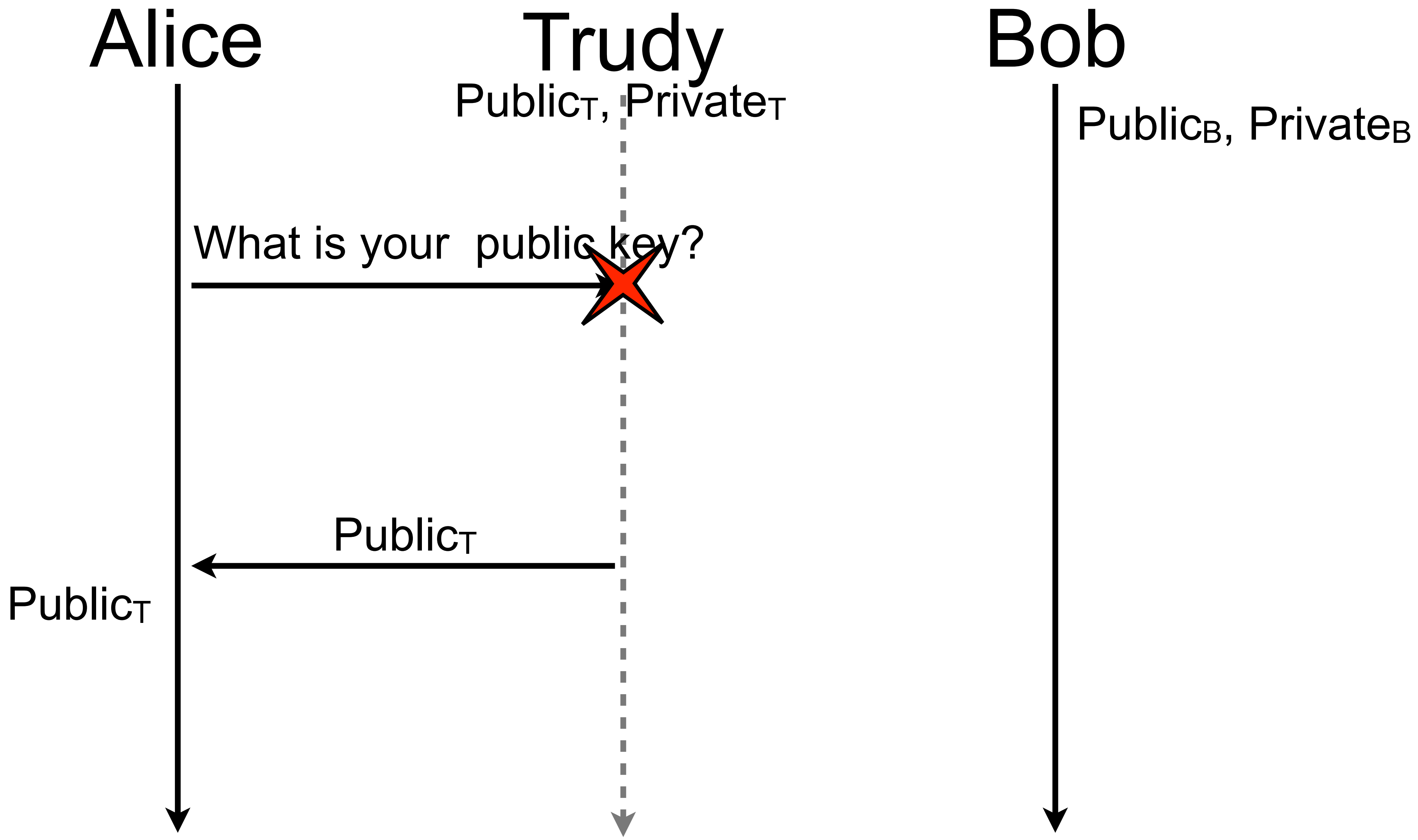
Public key infrastructure

- How to safely obtain Bob’s public key?



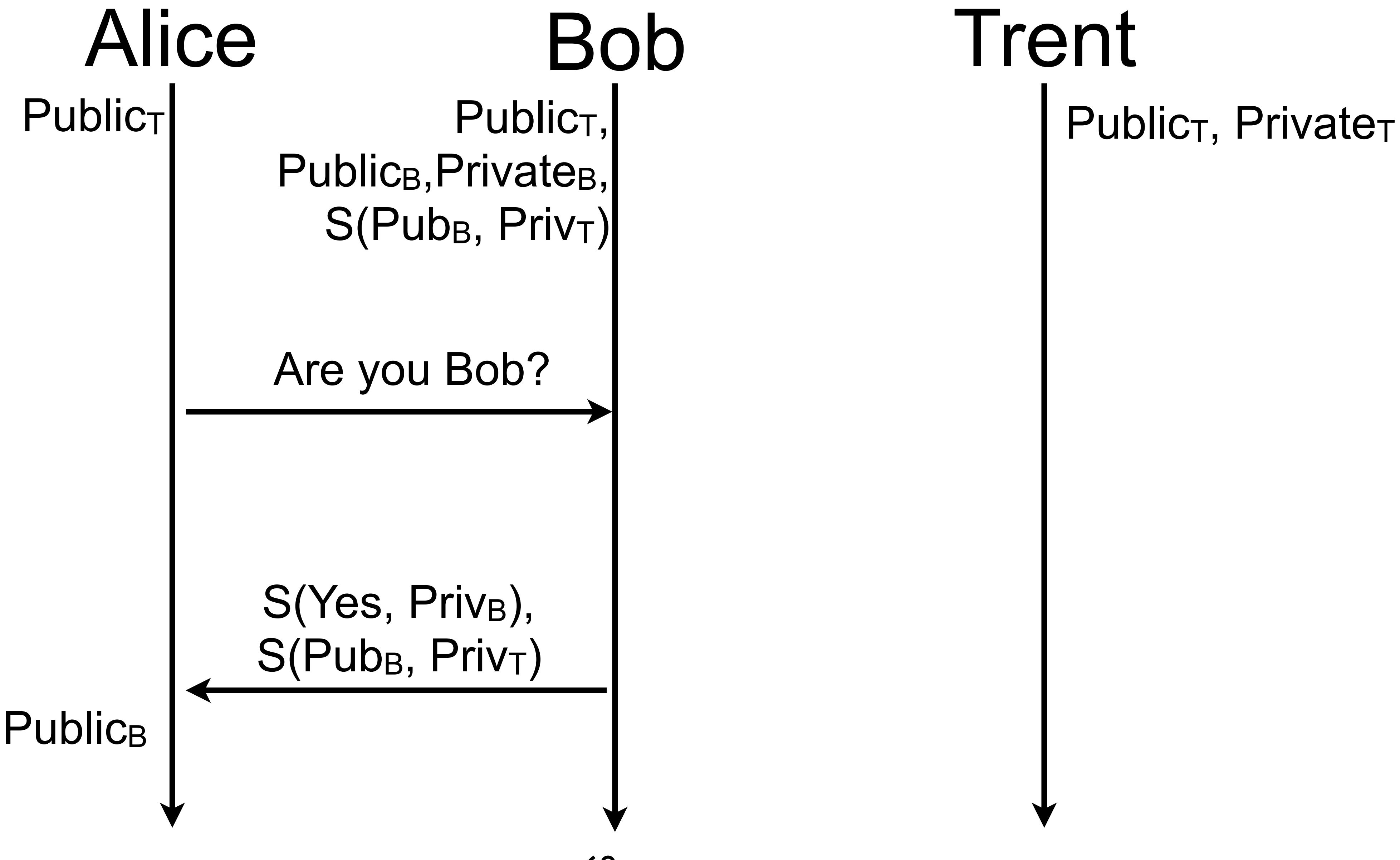
Public key infrastructure (contd.)

- Trudy can send a forged key


Alice
Trudy
Bob
Public_T, Private_T
Public_B, Private_B
What is your public key?
Public_T
Public_T

Public key infrastructure (contd.)

- Alice and Bob trust a third party (e.g., Trent) for authentication



Public key infrastructure (contd.)

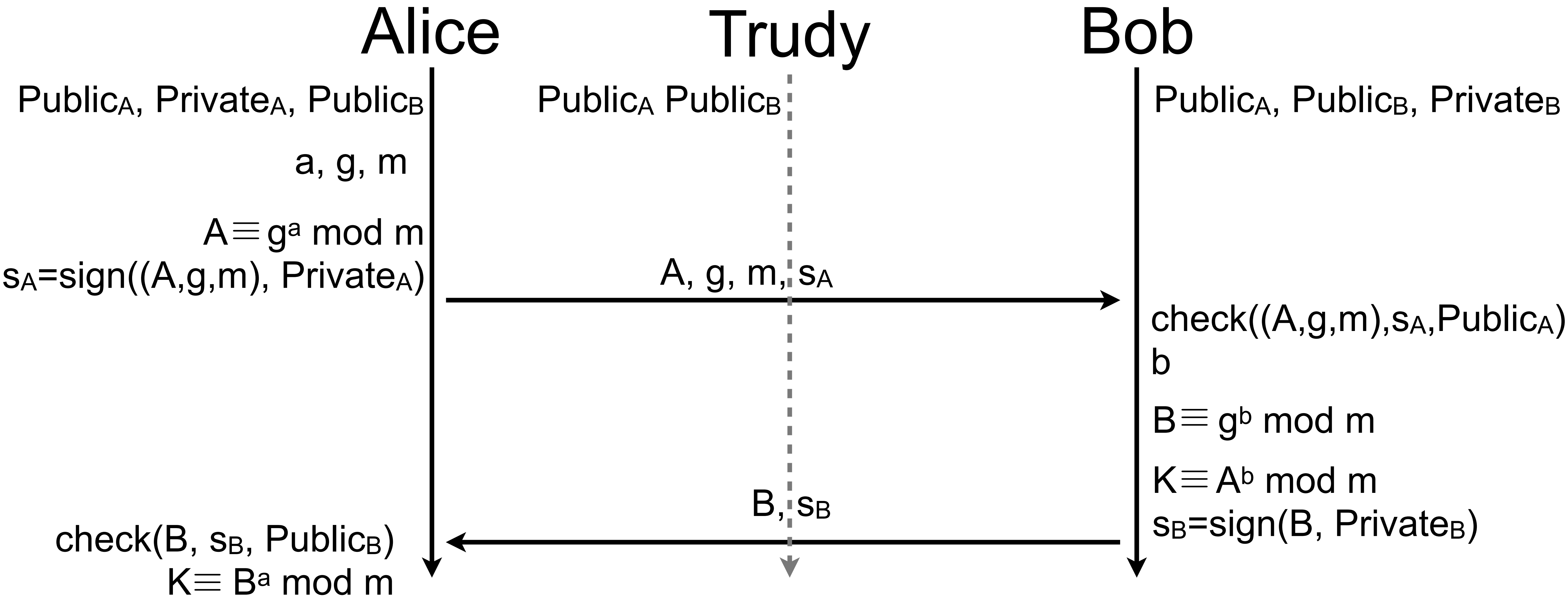
- Practically, Bob sends a certificate (e.g., X.509), not only its public key and signature
- A certificate provides many information to be able to correctly identify and authenticate its subject (e.g., Bob)
  - the subject name and organization
  - the subject public key (and type)
  - the issuer name and organization
  - the certificate validity time (valid not before and not after)
  - the certificate signature and type, signature made by the issuer of the certificate
  - ...

Public key infrastructure (contd.)

- Certificates are issued once and valid during a given time period, whatever the number of time it is used
- What if the subjects leaves its organization? The private key of the subject is stolen? The private key of the issuer is stolen?
- When a certified key is compromised, the certificate is revoked
  - the issuer maintains the list of revoked certificates
  - that should be checked by the client.

Diffie-Hellman key exchange (the return)

- Trudy cannot perform her attack anymore



Alice

Trudy

Bob

Public_A, Private_A, Public_B

Public_A Public_B

Public_A, Public_B, Private_B

a, g, m

$A \equiv g^a \bmod m$

$s_A = \text{sign}((A,g,m), \text{Private}_A)$

A, g, m, s_A

$\text{check}((A,g,m), s_A, \text{Public}_A)$

b

$B \equiv g^b \bmod m$

$K \equiv A^b \bmod m$

$s_B = \text{sign}(B, \text{Private}_B)$

B, s_B

$\text{check}(B, s_B, \text{Public}_B)$

$K \equiv B^a \bmod m$

Problem solved?

- fill me
- fill me
- fill me

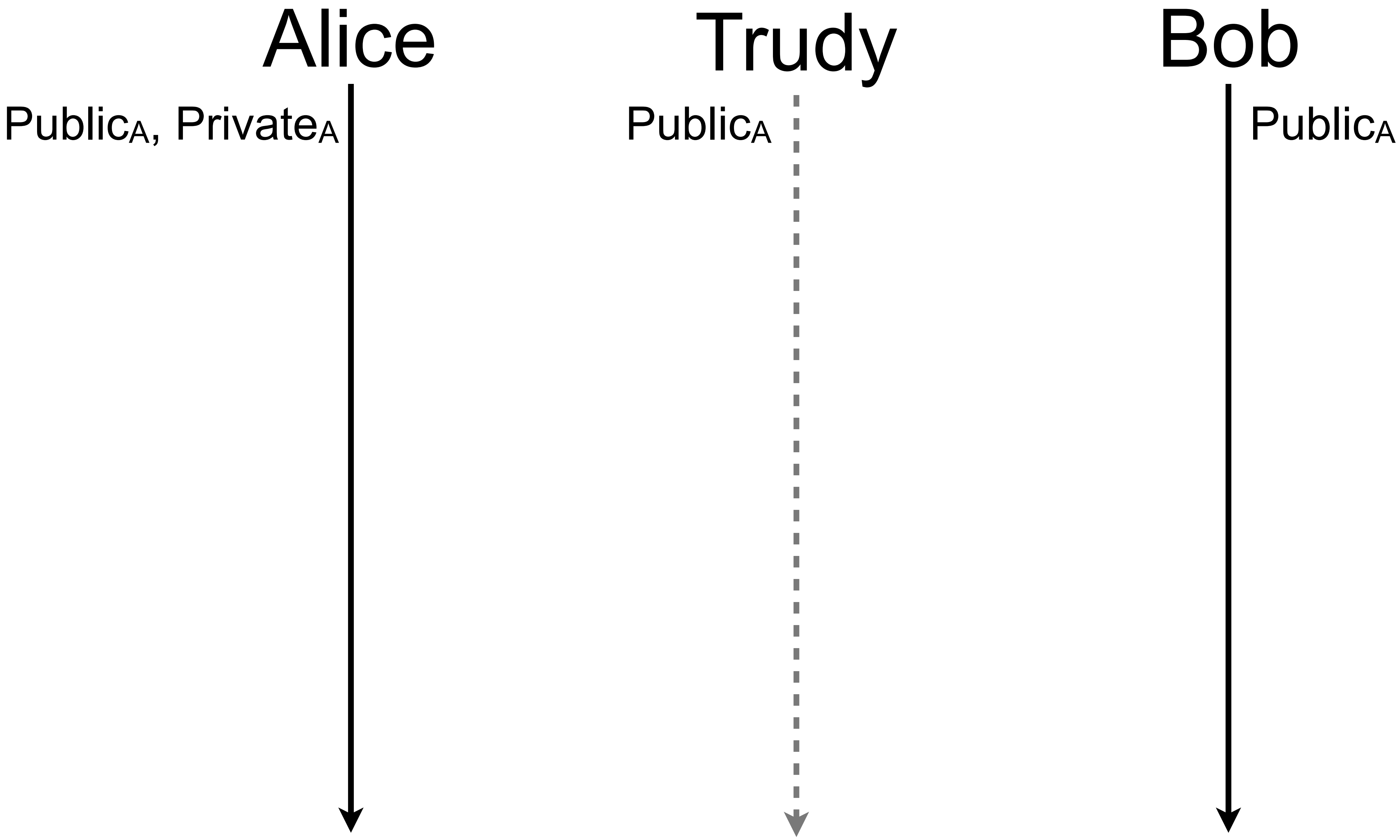
Problem solved?

- fill me
- fill me
- fill me

Replay attacks are still possible!

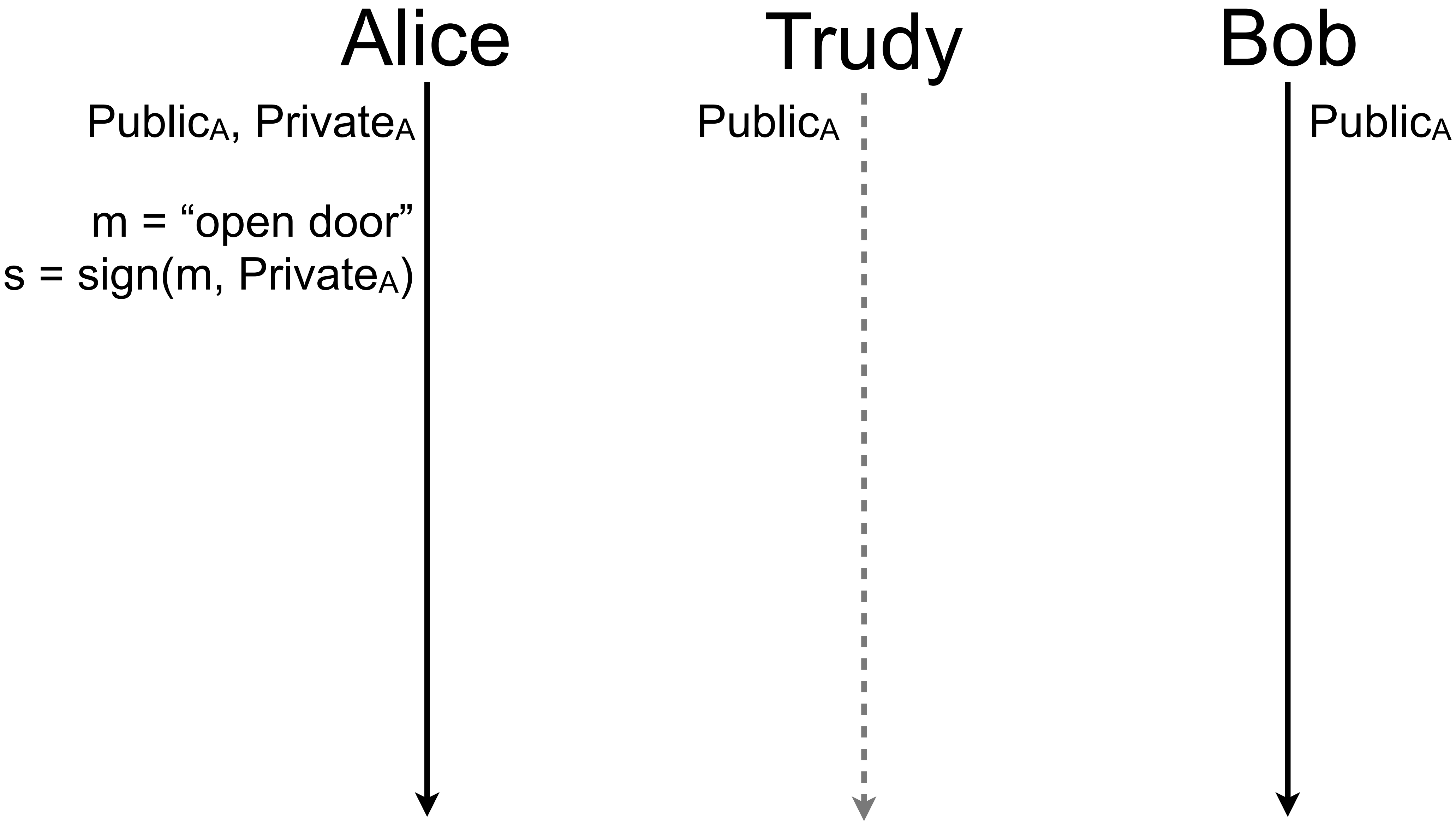
Nonce

- Trudy can replay a message



Nonce

- Trudy can replay a message



Alice

Trudy

Bob

Public$_A$, Private$_A$

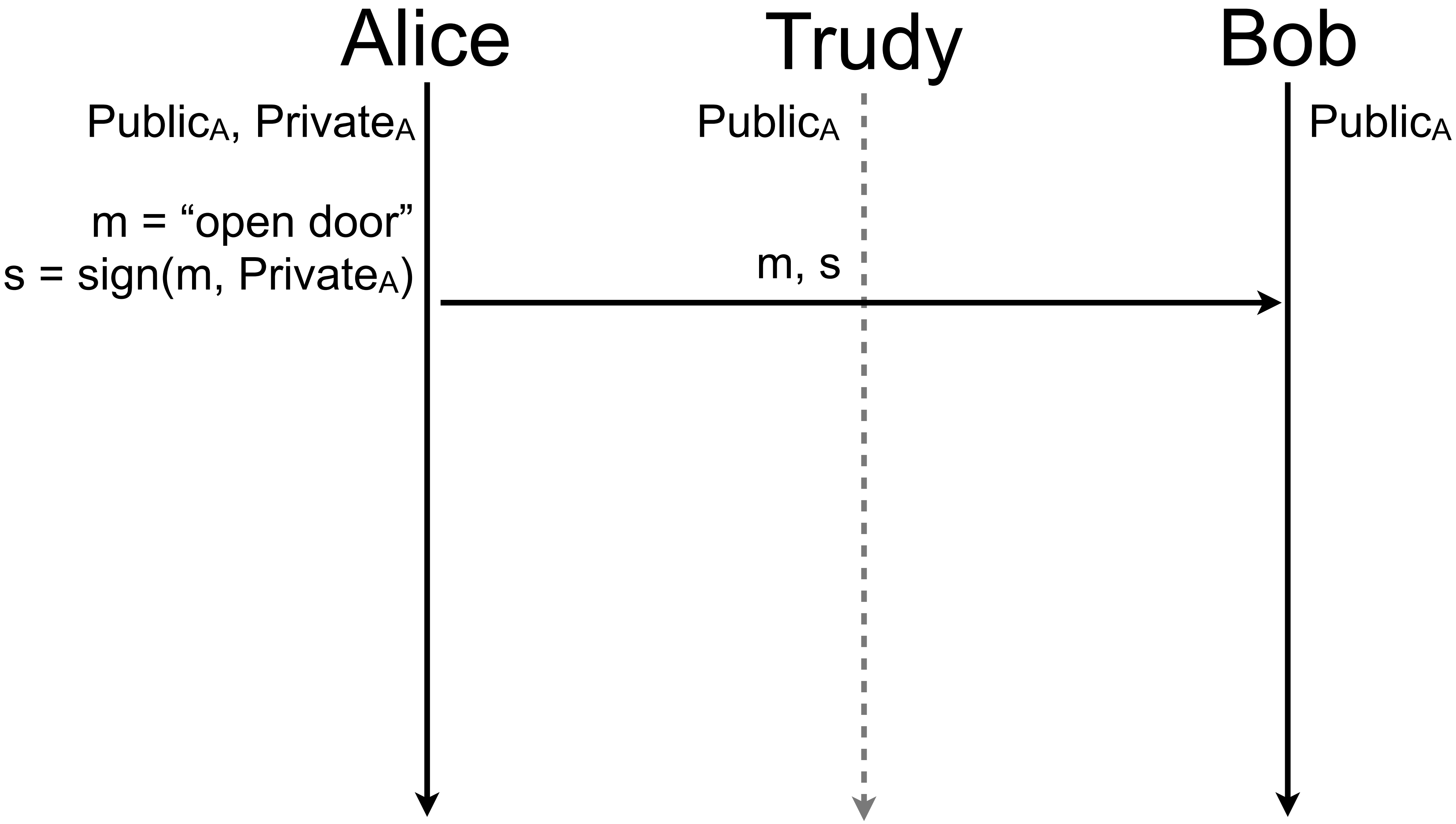
Public$_A$

Public$_A$

m = “open door”
s = sign(m, Private$_A$)

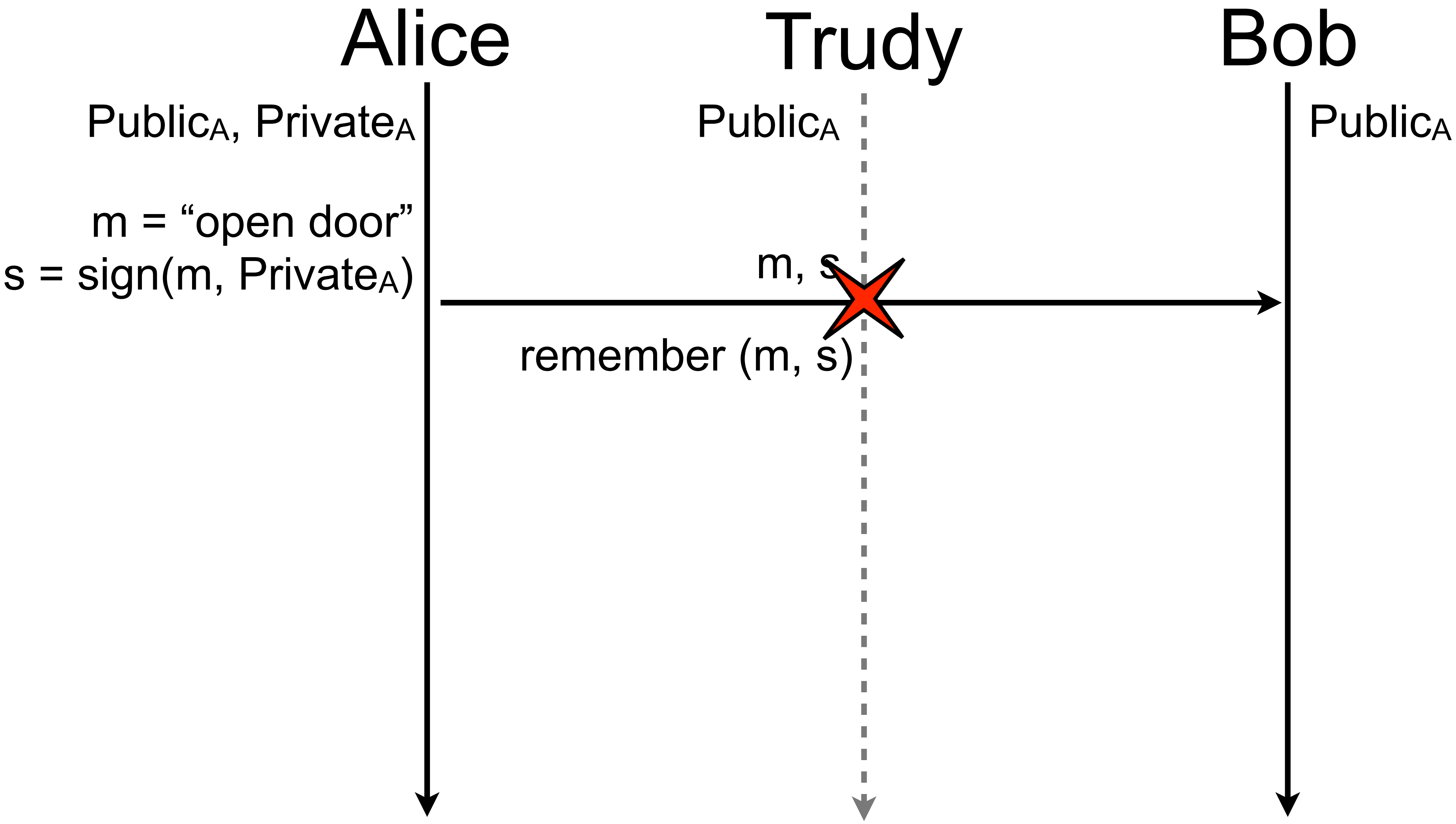
Nonce

- Trudy can replay a message



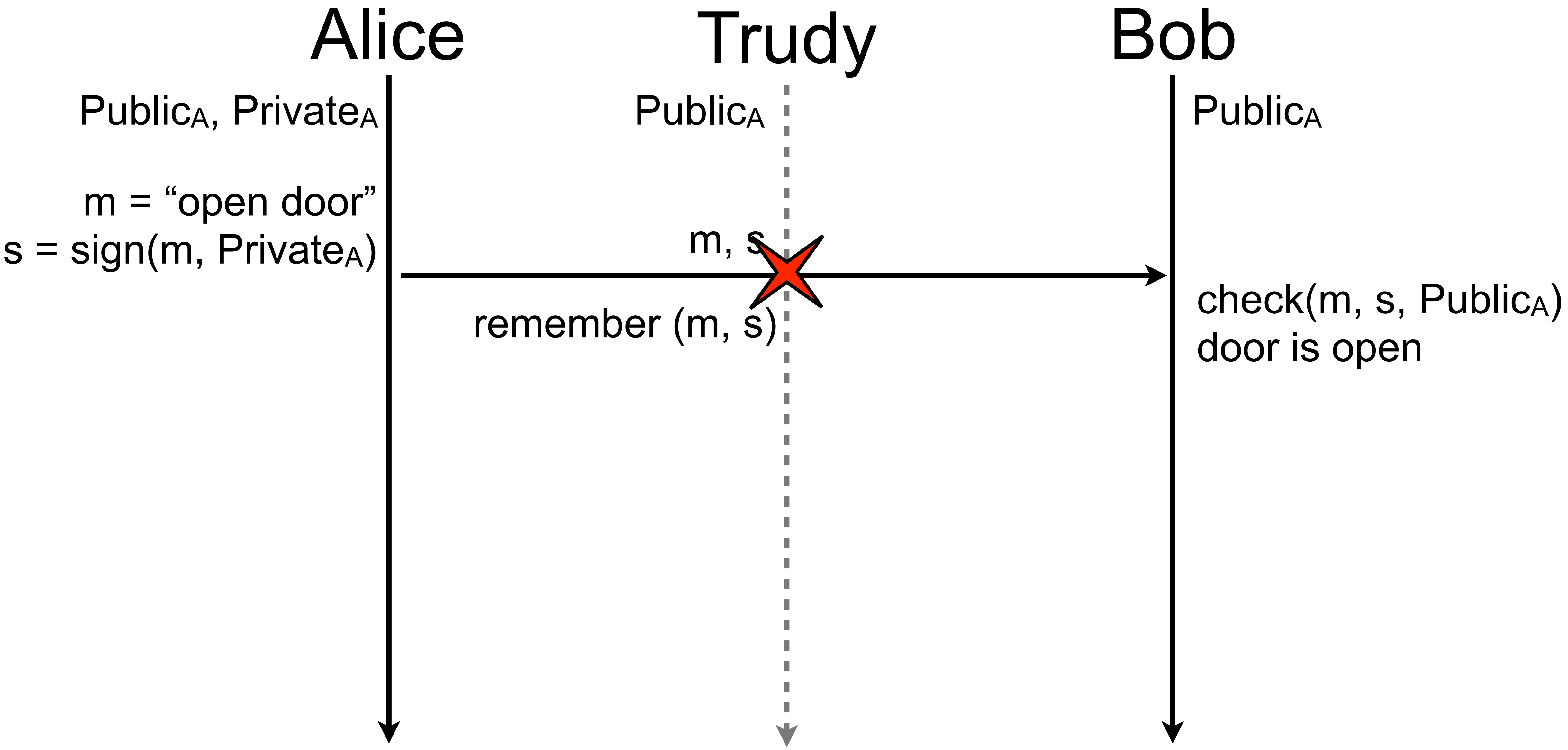
Nonce

- Trudy can replay a message



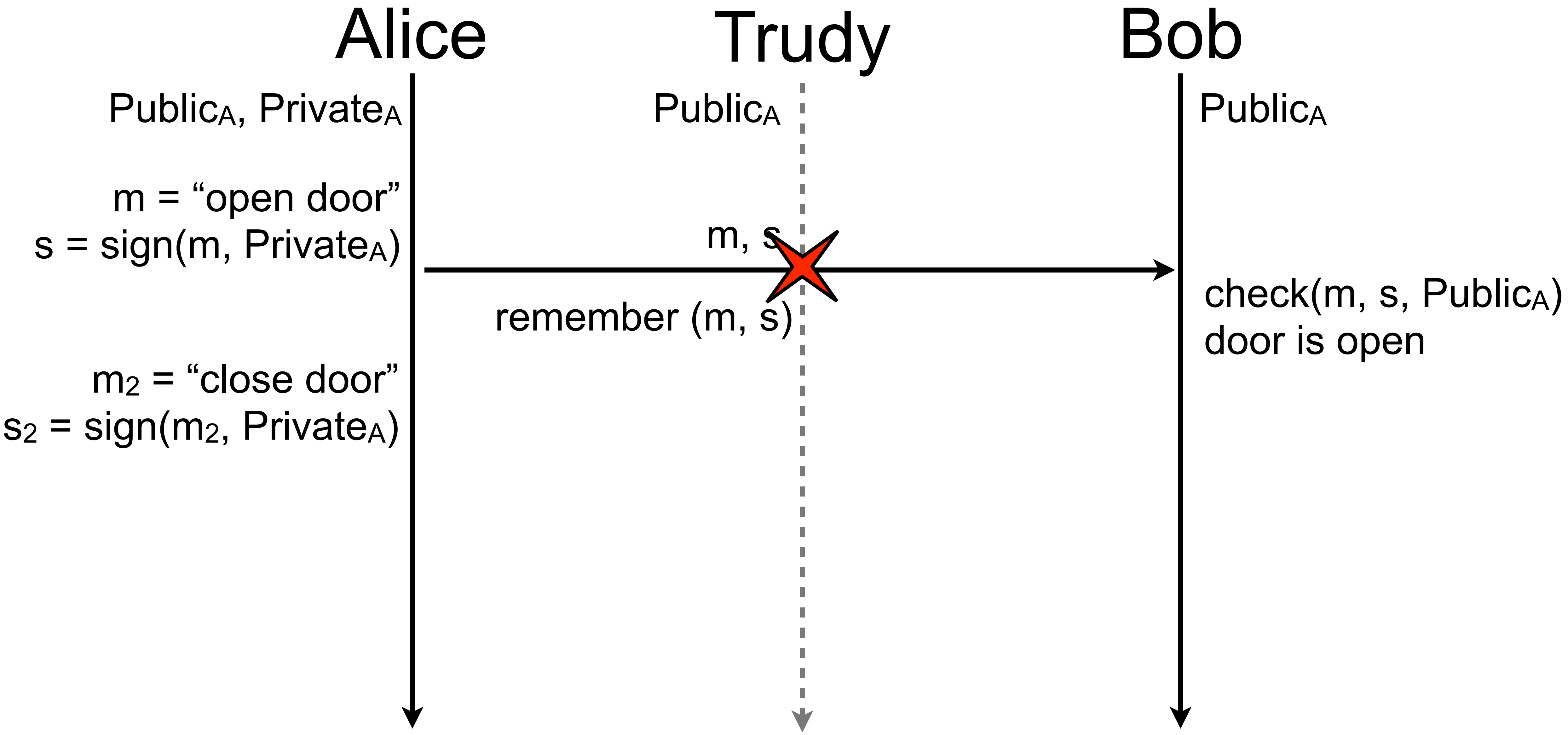
Nonce

- Trudy can replay a message



Nonce

- Trudy can replay a message



Alice

Trudy

Bob

$Public_A$, $Private_A$

$Public_A$

$Public_A$

m = “open door”
s = sign(m, $Private_A$)

m, s

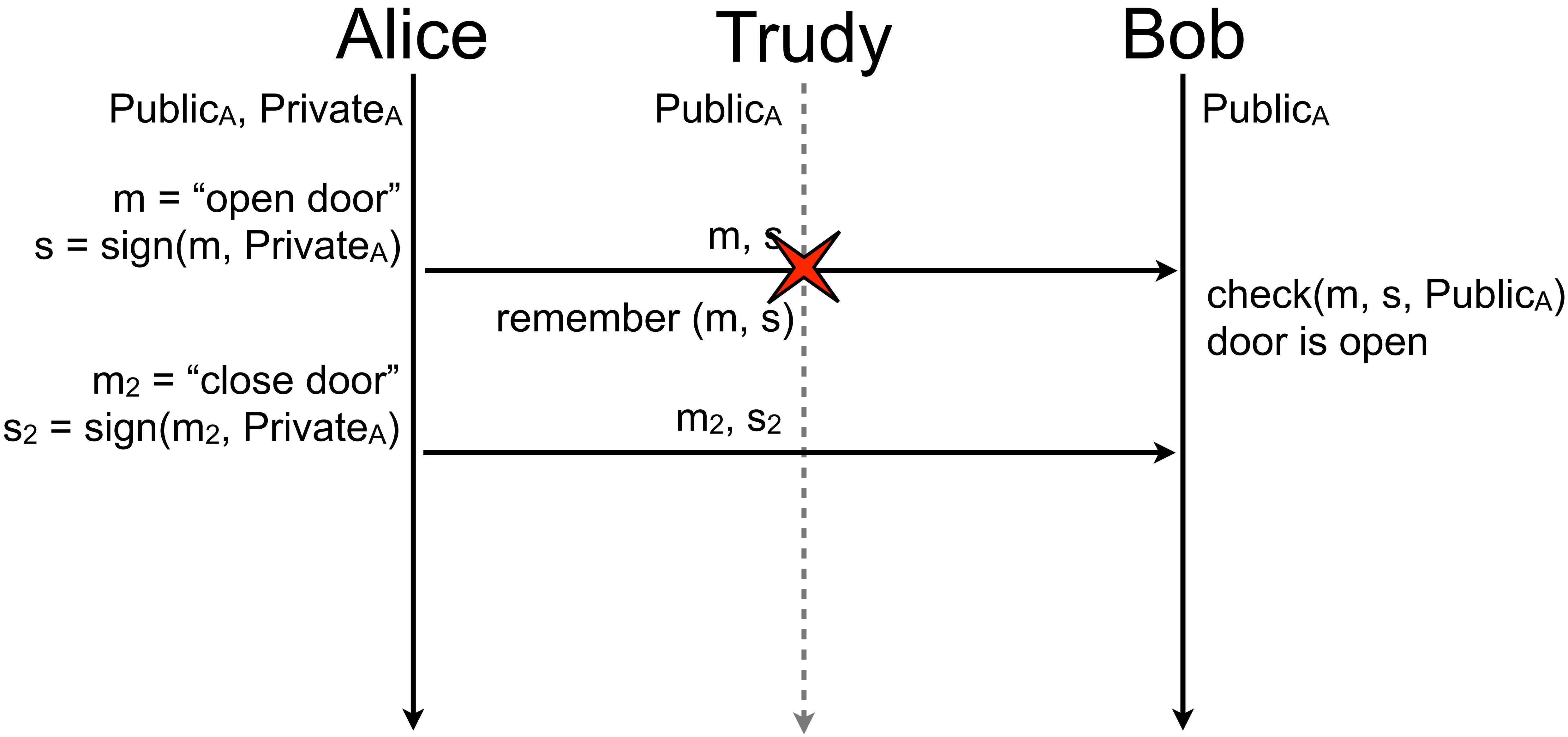
remember (m, s)

check(m, s, $Public_A$)
door is open

m_2 = “close door”
s_2 = sign(m_2, $Private_A$)

Nonce

- Trudy can replay a message



Alice

Trudy

Bob

Public_A, Private_A

Public_A

Public_A

m = "open door"

s = sign(m, Private_A)

m, s

remember (m, s)

check(m, s, Public_A)

door is open

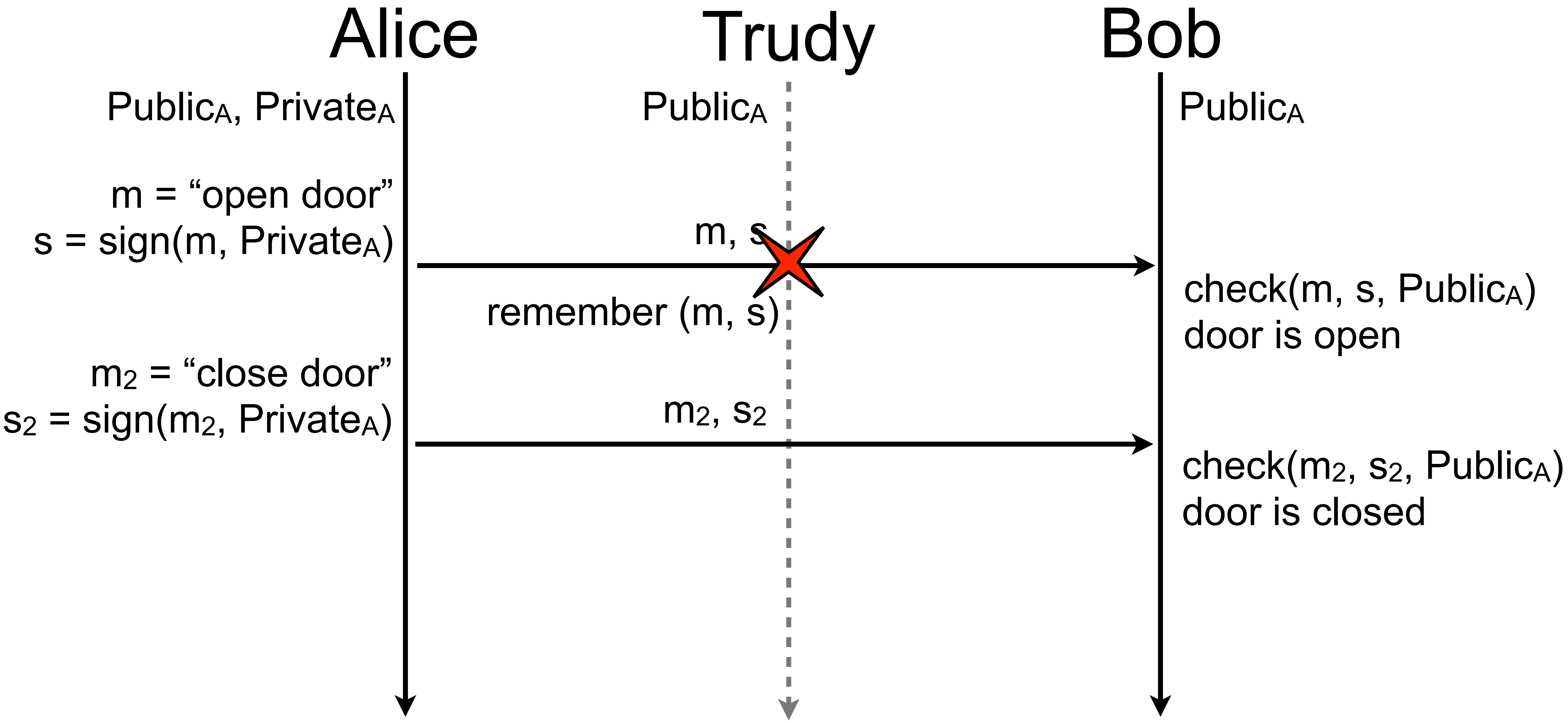
m_2 = "close door"

s_2 = sign(m_2, Private_A)

m_2, s_2

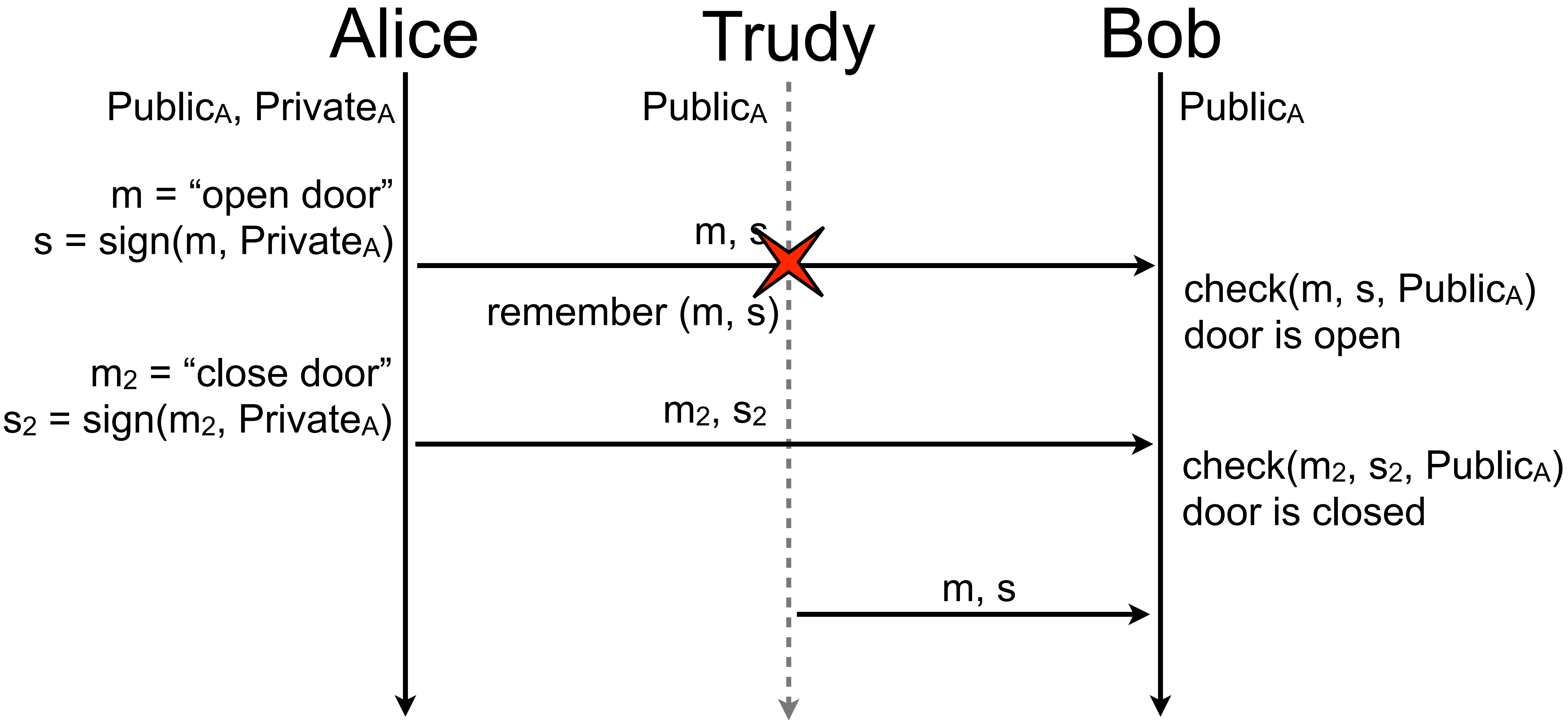
Nonce

- Trudy can replay a message



Nonce

- Trudy can replay a message



Alice

Trudy

Bob

Public$_A$, Private$_A$

Public$_A$

Public$_A$

m = “open door”
s = sign(m, Private$_A$)

m, s

remember (m, s)

check(m, s, Public$_A$)
door is open

m_2 = “close door”
s_2 = sign(m_2, Private$_A$)

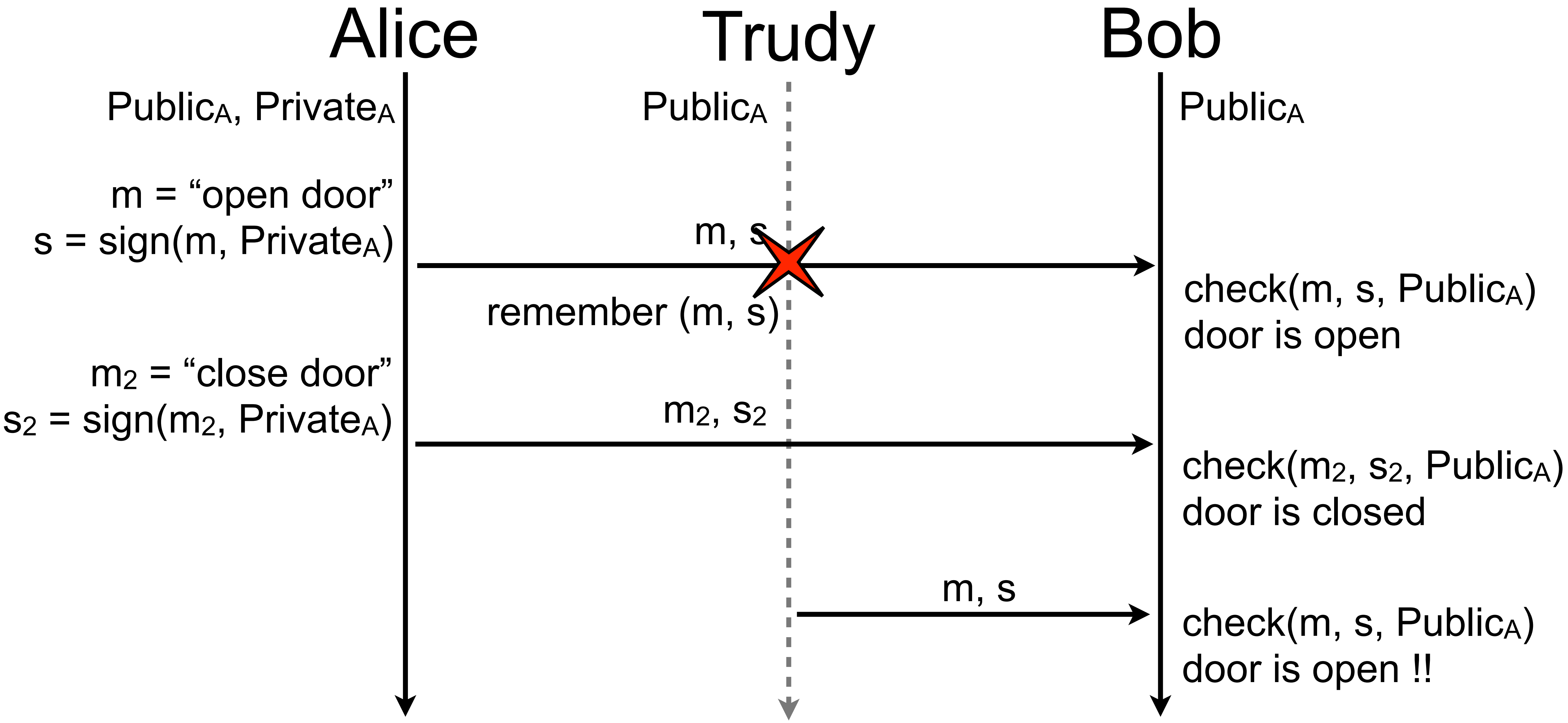
m_2, s_2

check(m_2, s_2, Public$_A$)
door is closed

m, s

Nonce

- Trudy can replay a message



Alice

Trudy

Bob

Public$_A$, Private$_A$

Public$_A$

Public$_A$

m = “open door”
s = sign(m, Private$_A$)

m, s

remember (m, s)

check(m, s, Public$_A$)
door is open

m_2 = “close door”
s_2 = sign(m_2, Private$_A$)

m_2, s_2

check(m_2, s_2, Public$_A$)
door is closed

m, s

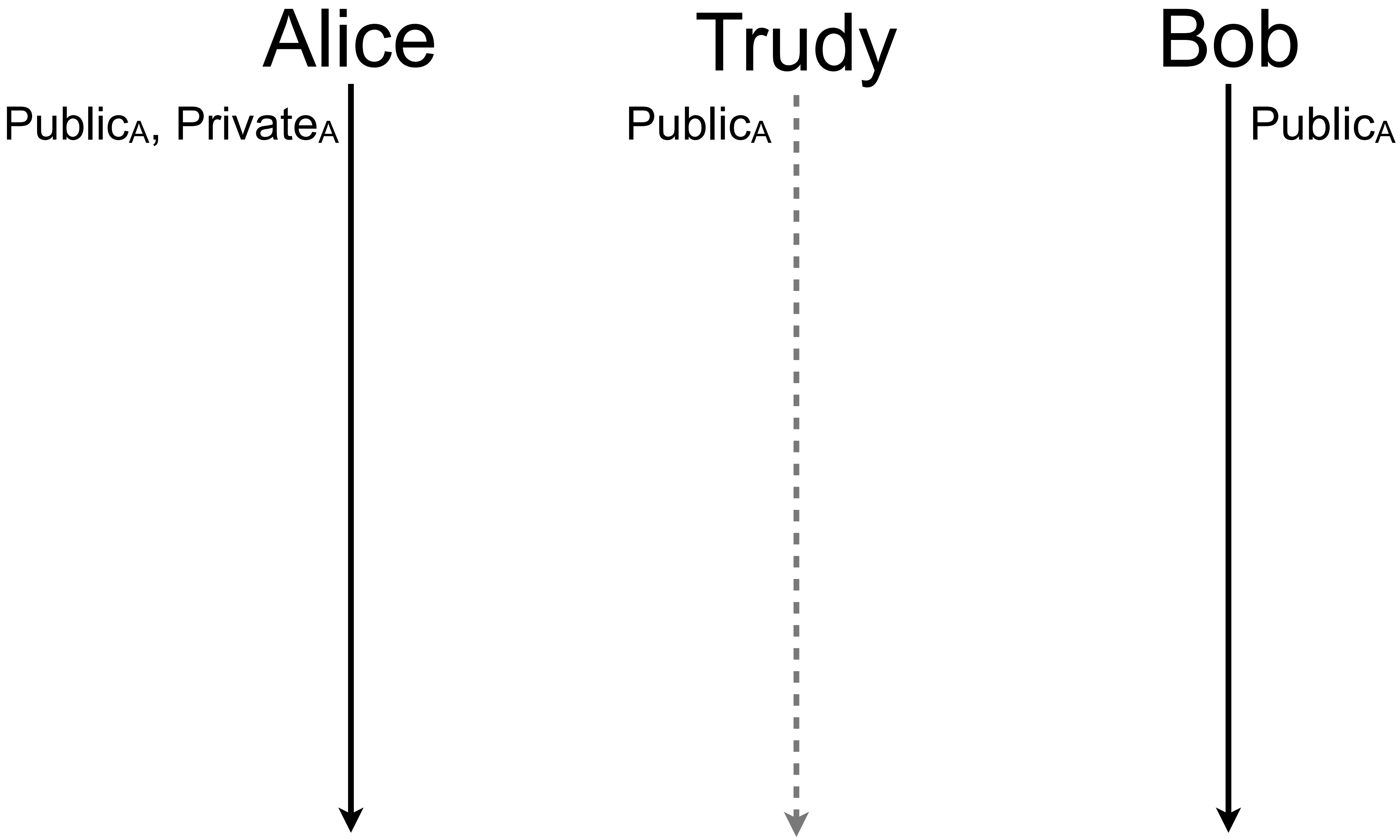
check(m, s, Public$_A$)
door is open !!

Nonce (contd.)

- A nonce is a number used only once
- Three general methods to create nonces
  - sequential number
    - increment after each use
    - keep it in non-volatile storage in case of reboot
  - timestamp
    - current time of the nonce generation
    - be sure clock is not going backward (e.g., winter time)
  - random number
    - low collision probability if the pseudo random number generator is good and random number is big enough (e.g., more than 128 bits)
- Nonce alone is rarely enough to have a good protection
  - not robust to eavesdropping or man-in-the-middle attack

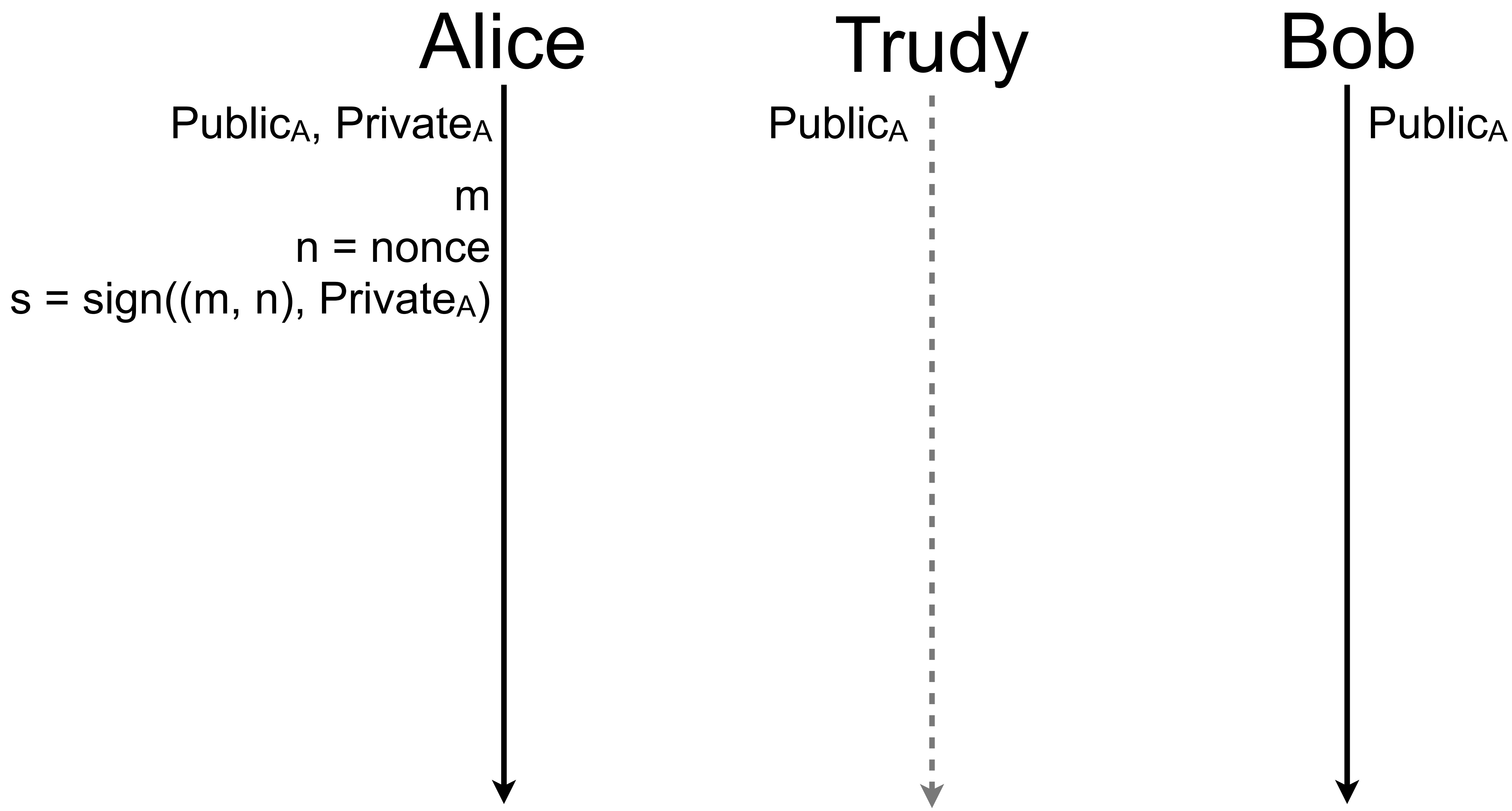
Nonce (contd.)

- Each message is make unique thanks to the nonce



Nonce (contd.)

- Each message is make unique thanks to the nonce



Alice

Trudy

Bob

Public_A, Private_A

m

n = nonce

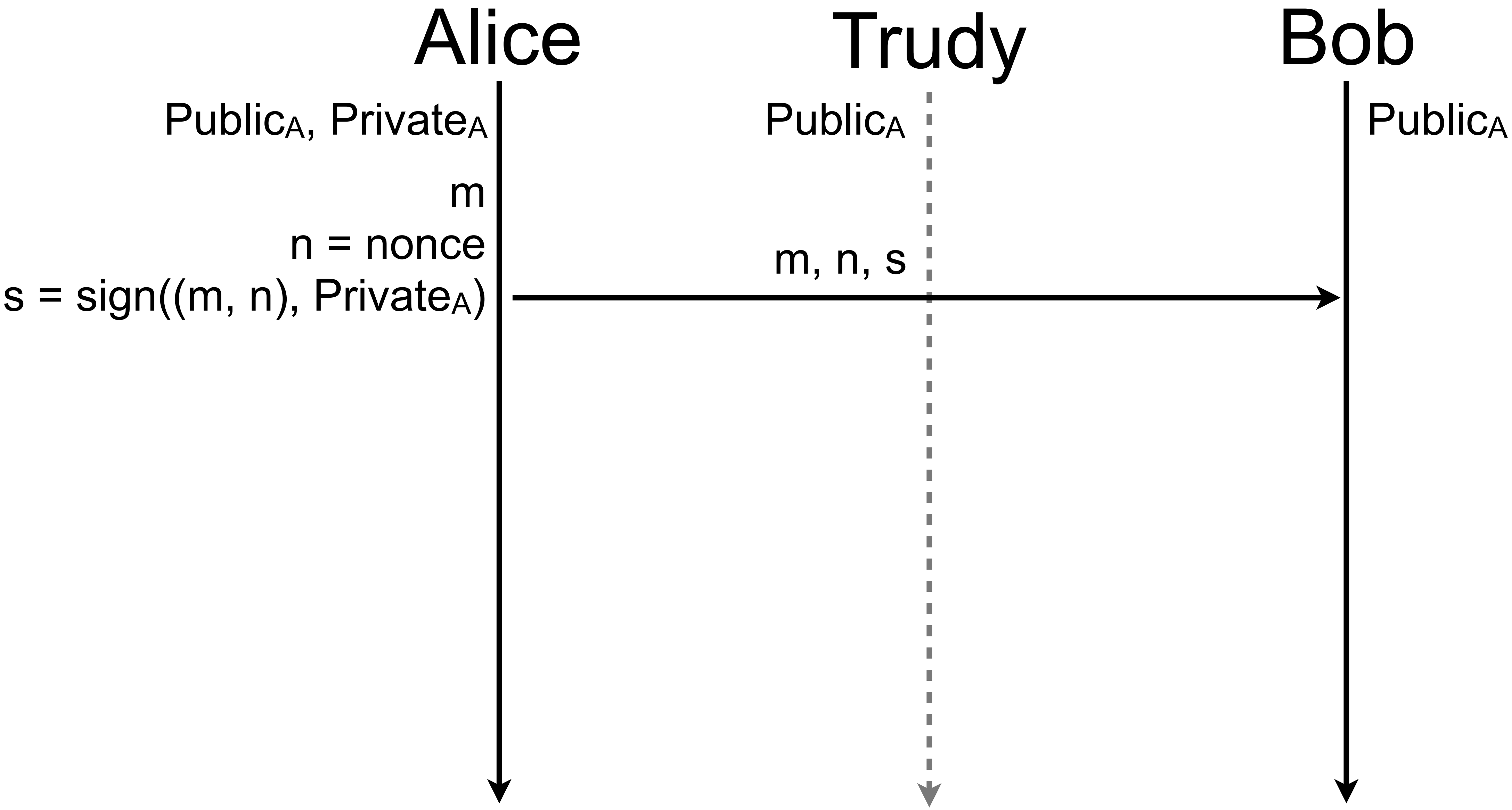
s = sign((m, n), Private_A)

Public_A

Public_A

Nonce (contd.)

- Each message is make unique thanks to the nonce



Alice

Trudy

Bob

Public_A, Private_A

Public_A

Public_A

m

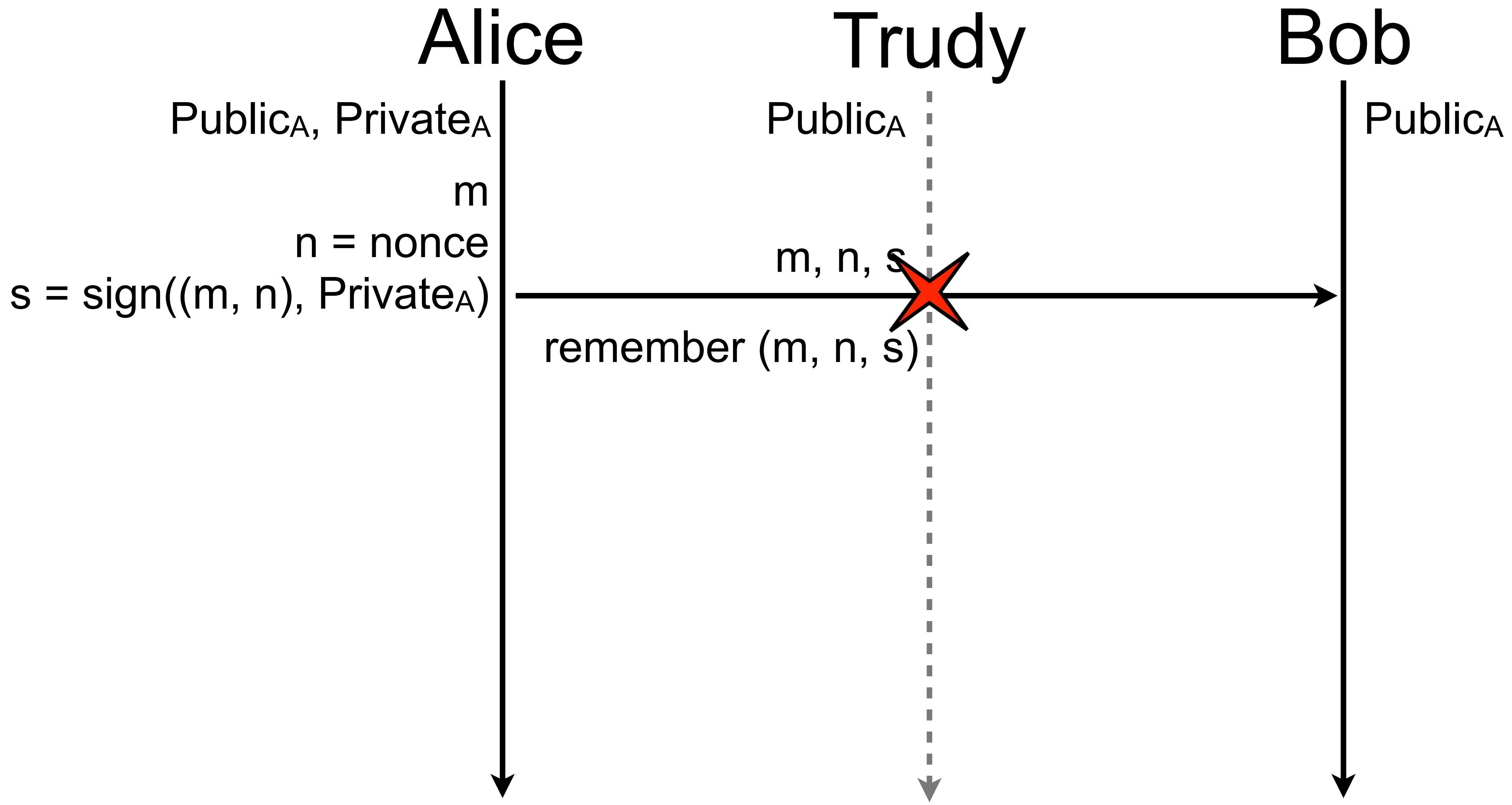
n = nonce

s = sign((m, n), Private_A)

m, n, s

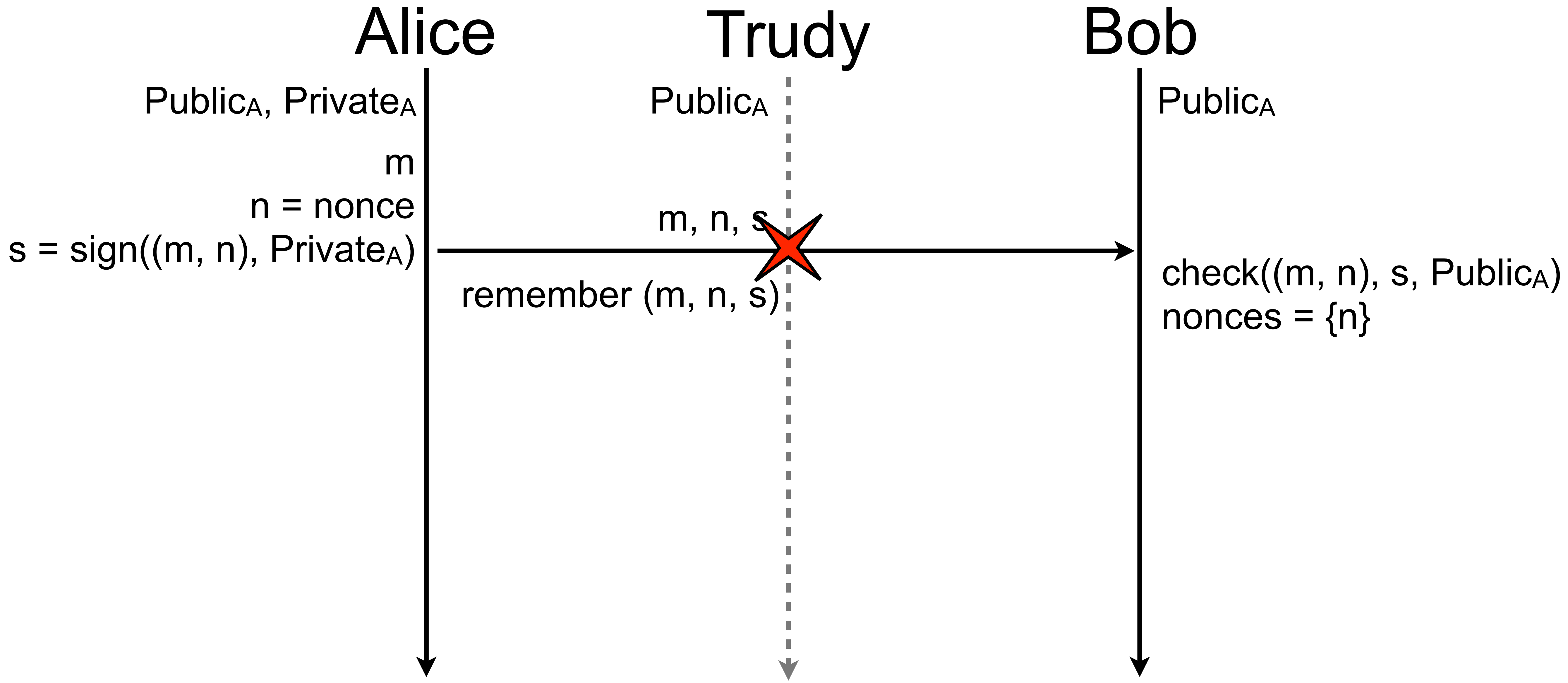
Nonce (contd.)

- Each message is make unique thanks to the nonce



Nonce (contd.)

- Each message is make unique thanks to the nonce



Alice

Trudy

Bob

$Public_A$, $Private_A$

$Public_A$

$Public_A$

m

n = nonce

s = sign((m, n), $Private_A$)

m, n, s

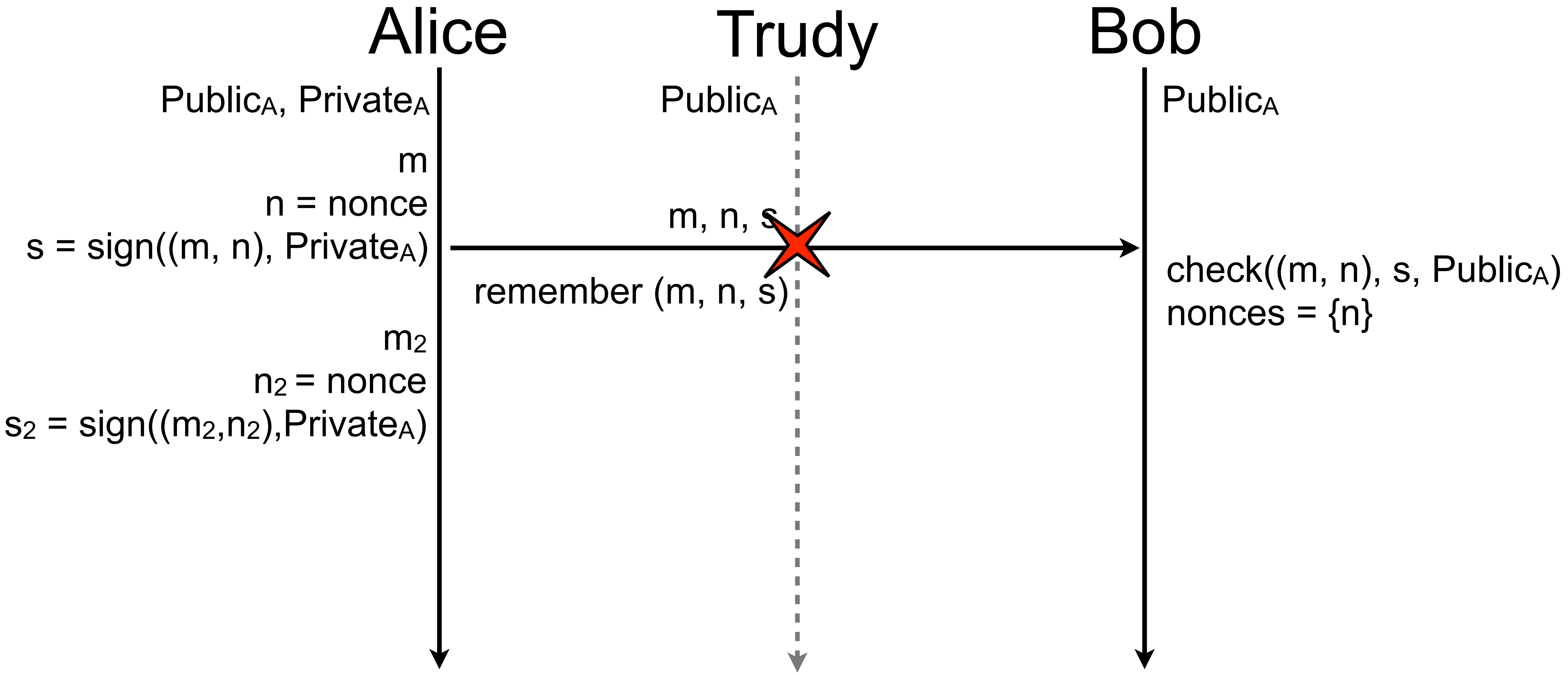
remember (m, n, s)

check((m, n), s, $Public_A$)

nonces = {n}

Nonce (contd.)

- Each message is make unique thanks to the nonce



Alice

Trudy

Bob

$Public_A$, $Private_A$

$Public_A$

$Public_A$

m

n = nonce

s = sign((m, n), $Private_A$)

m, n, s

remember (m, n, s)

check((m, n), s, $Public_A$)

nonces = {n}

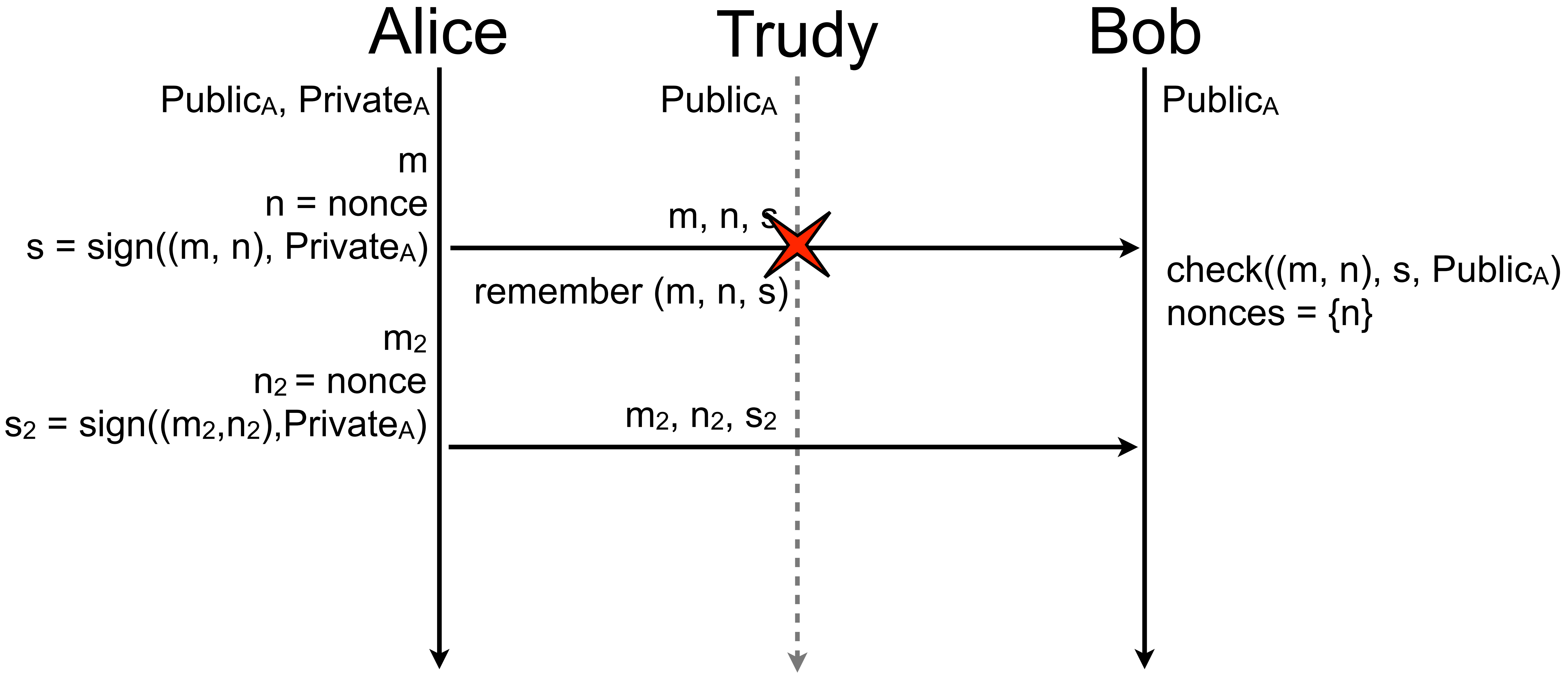
m_2

n_2 = nonce

s_2 = sign((m_2,n_2),$Private_A$)

Nonce (contd.)

- Each message is make unique thanks to the nonce



Alice

Trudy

Bob

$Public_A$, $Private_A$

$Public_A$

$Public_A$

m

n = nonce

s = sign((m, n), $Private_A$)

m, n, s

remember (m, n, s)

check((m, n), s, $Public_A$)

nonces = {n}

m_2

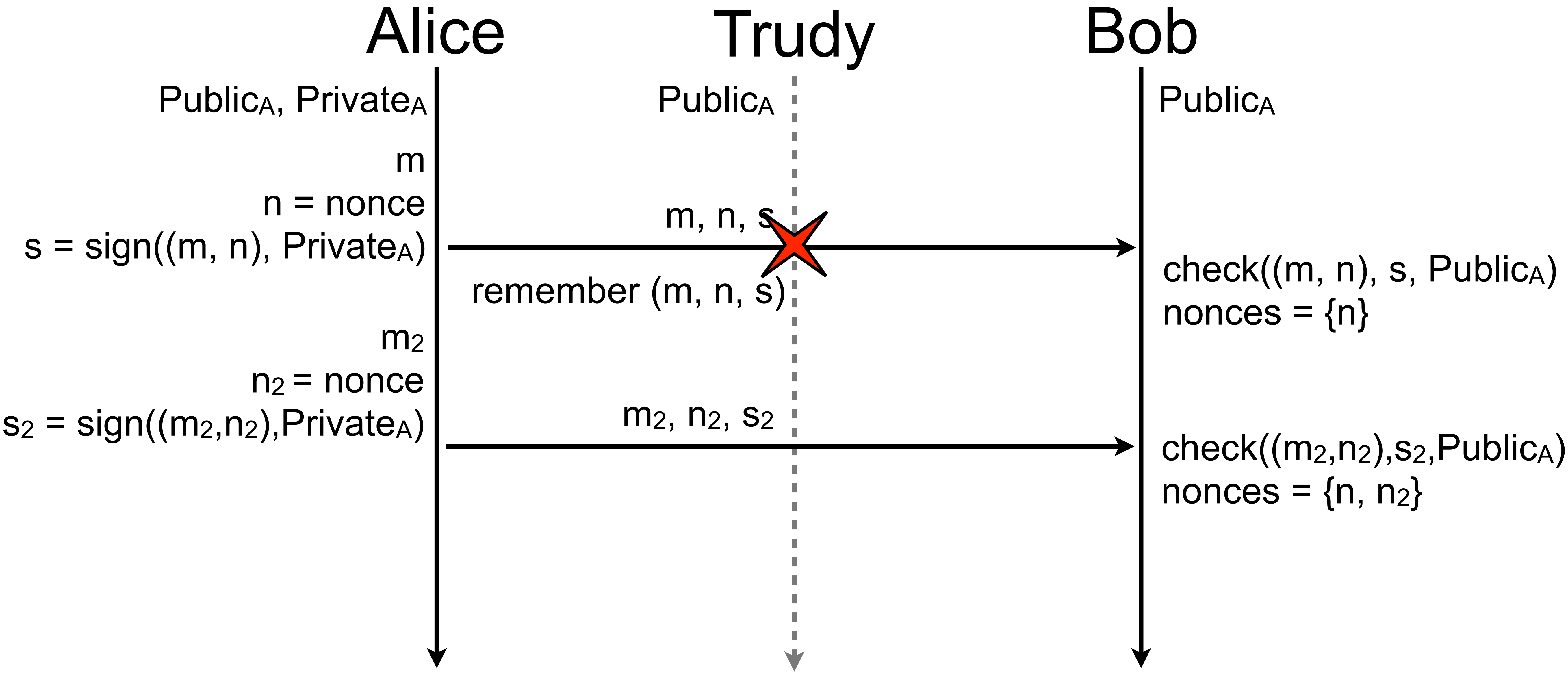
n_2 = nonce

s_2 = sign((m_2,n_2),$Private_A$)

m_2, n_2, s_2

Nonce (contd.)

- Each message is make unique thanks to the nonce



Alice

Trudy

Bob

$Public_A$, $Private_A$

$Public_A$

$Public_A$

m

n = nonce

s = sign((m, n), $Private_A$)

m, n, s

remember (m, n, s)

check((m, n), s, $Public_A$)

nonces = {n}

m_2

n_2 = nonce

s_2 = sign((m_2,n_2),$Private_A$)

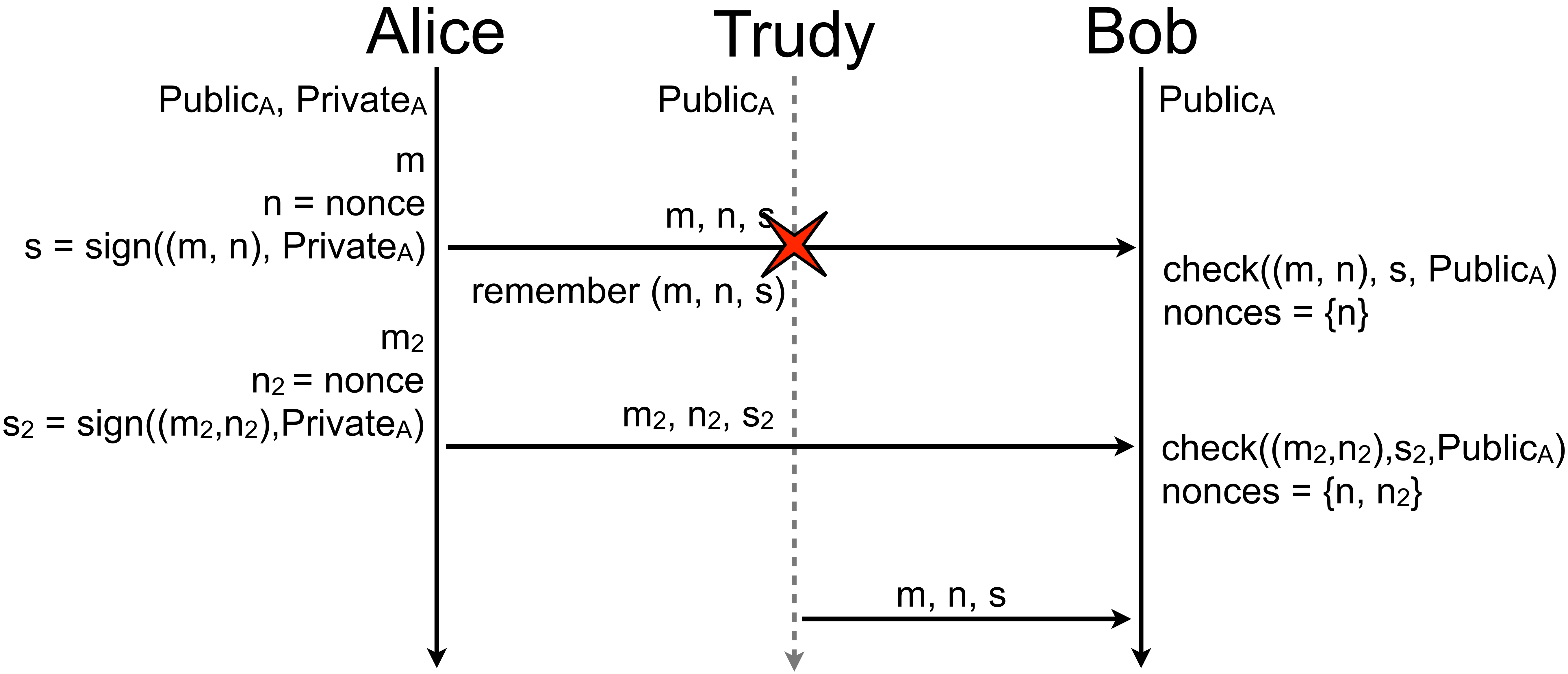
m_2, n_2, s_2

check((m_2,n_2),s_2,$Public_A$)

nonces = {n, n_2}

Nonce (contd.)

- Each message is make unique thanks to the nonce



Alice

Trudy

Bob

$Public_A$, $Private_A$

$Public_A$

$Public_A$

m

n = nonce

s = sign((m, n), $Private_A$)

m, n, s

remember (m, n, s)

check((m, n), s, $Public_A$)

nonces = {n}

m_2

n_2 = nonce

s_2 = sign((m_2,n_2),$Private_A$)

m_2, n_2, s_2

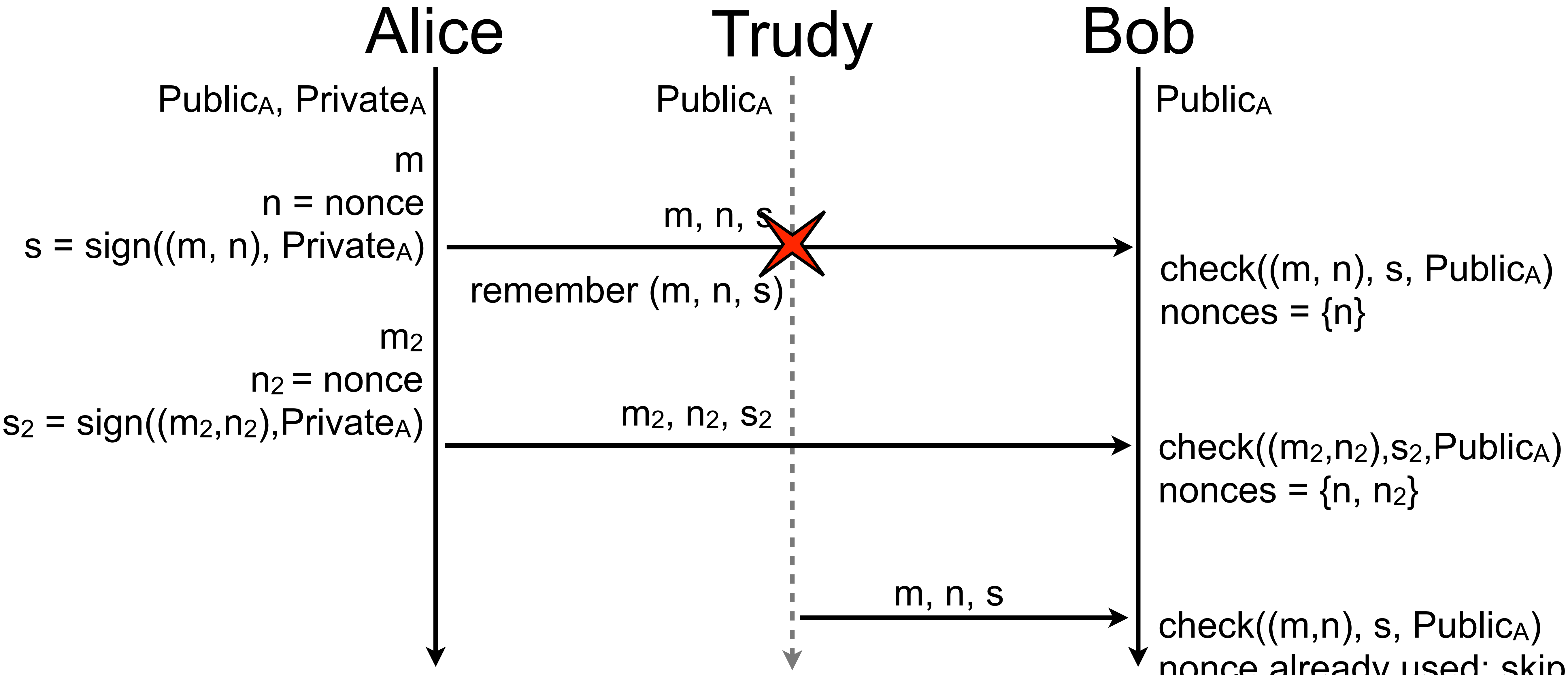
check((m_2,n_2),s_2,$Public_A$)

nonces = {n, n_2}

m, n, s

Nonce (contd.)

- Each message is make unique thanks to the nonce



Alice

Trudy

Bob

$Public_A$, $Private_A$

$Public_A$

$Public_A$

m

n = nonce

s = sign((m, n), $Private_A$)

m, n, s

remember (m, n, s)

check((m, n), s, $Public_A$)

nonces = {n}

m_2

n_2 = nonce

s_2 = sign((m_2,n_2),$Private_A$)

m_2, n_2, s_2

check((m_2,n_2),s_2,$Public_A$)

nonces = {n, n_2}

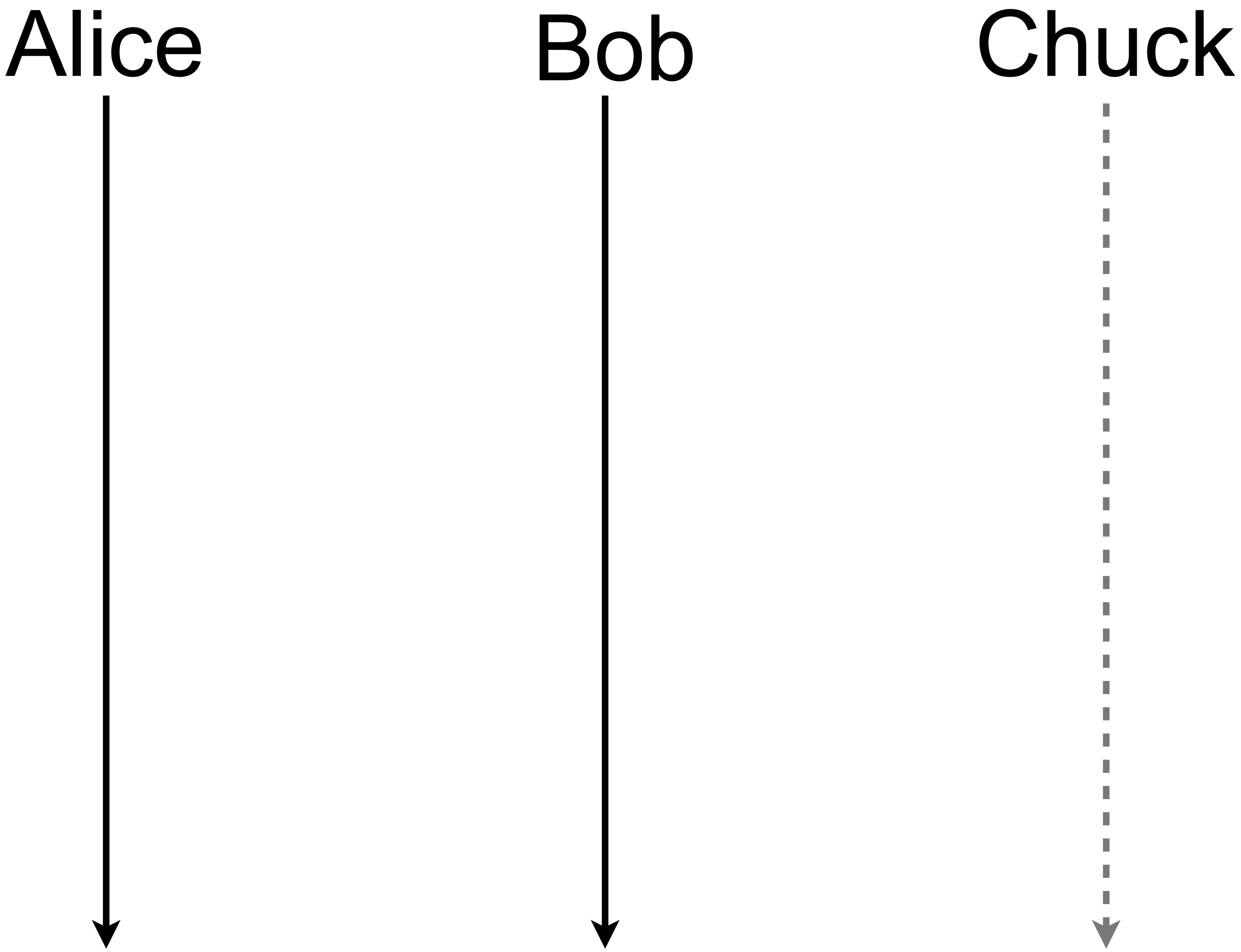
m, n, s

check((m,n), s, $Public_A$)

nonce already used: skip

Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



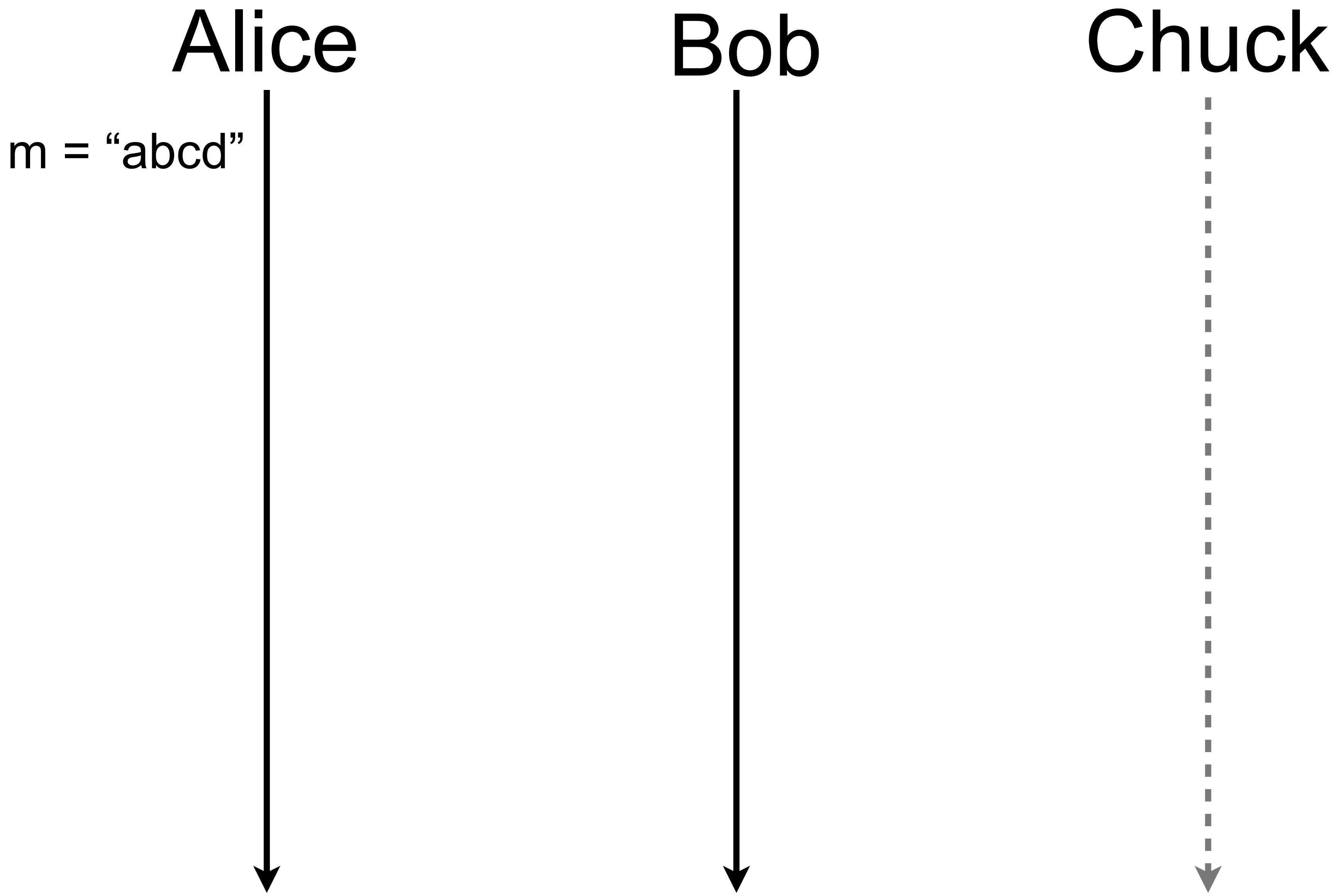
Alice

Bob

Chuck

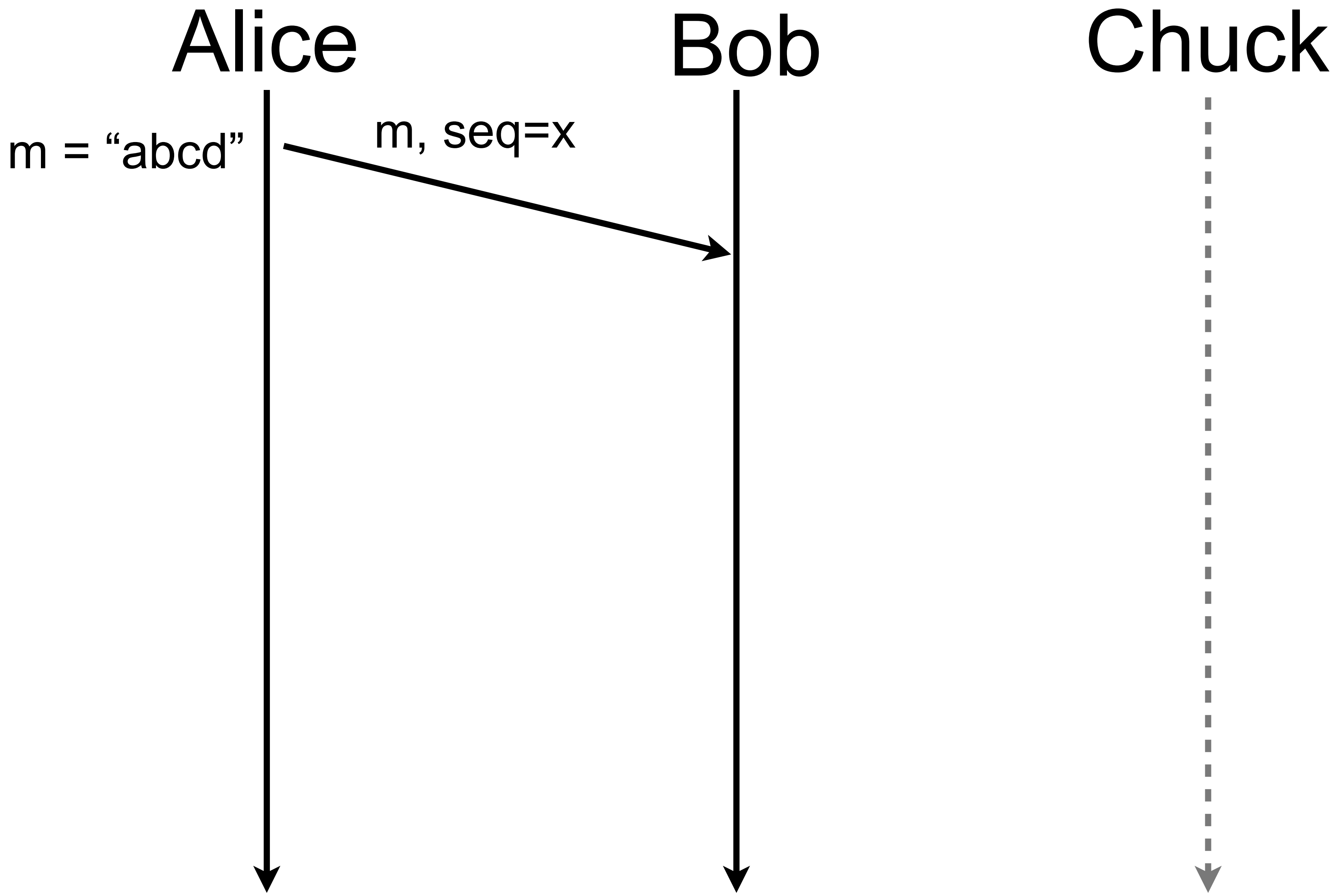
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



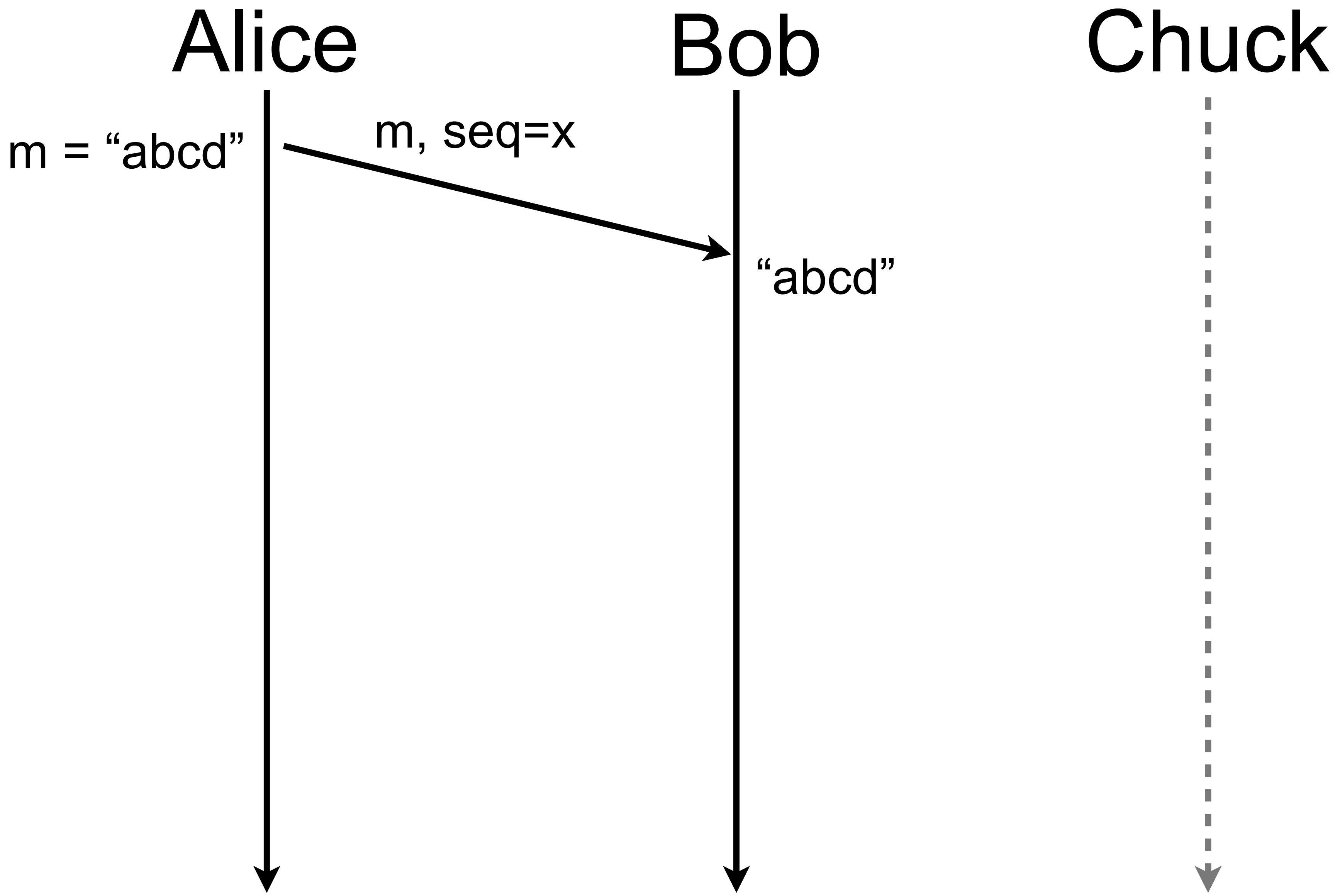
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



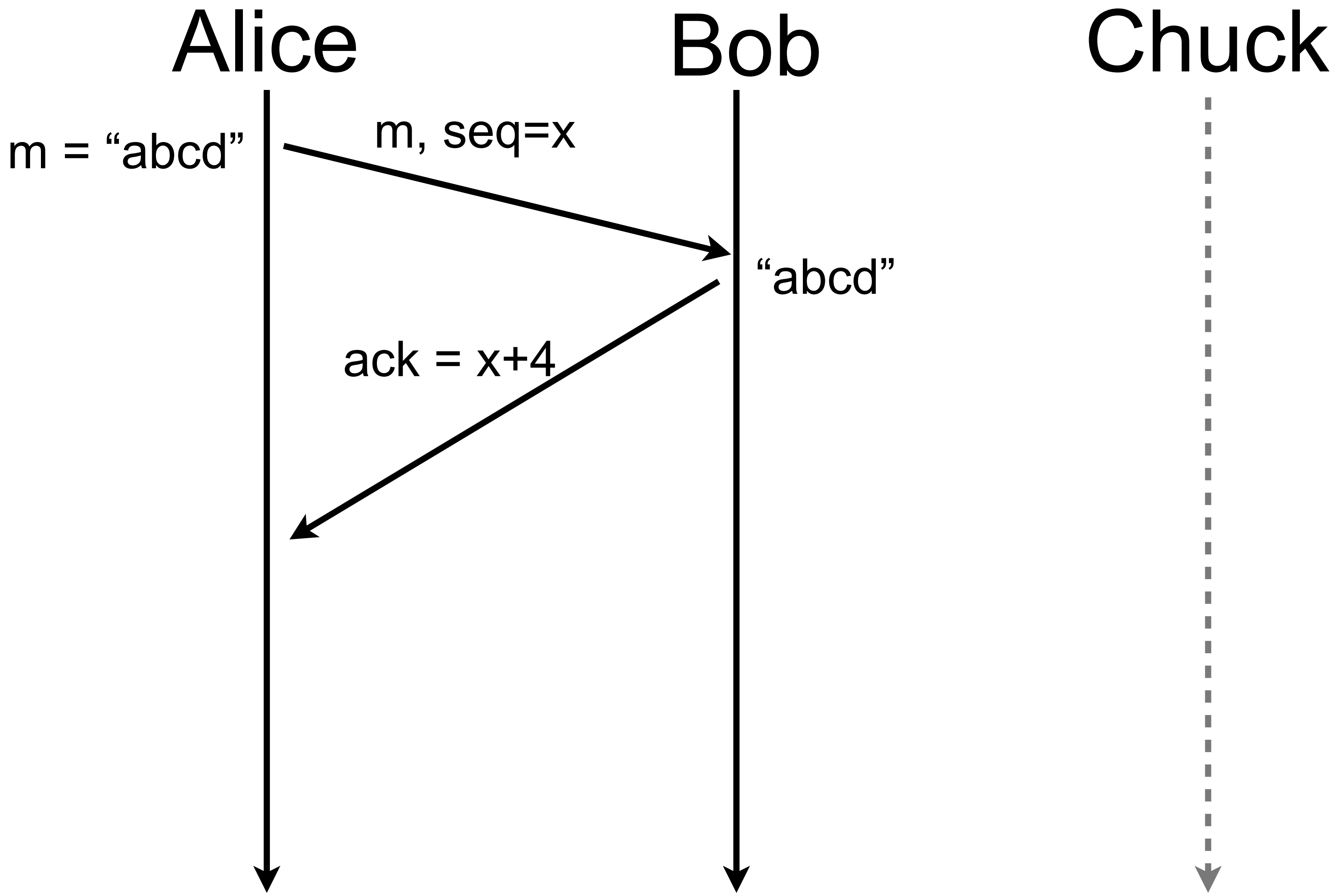
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



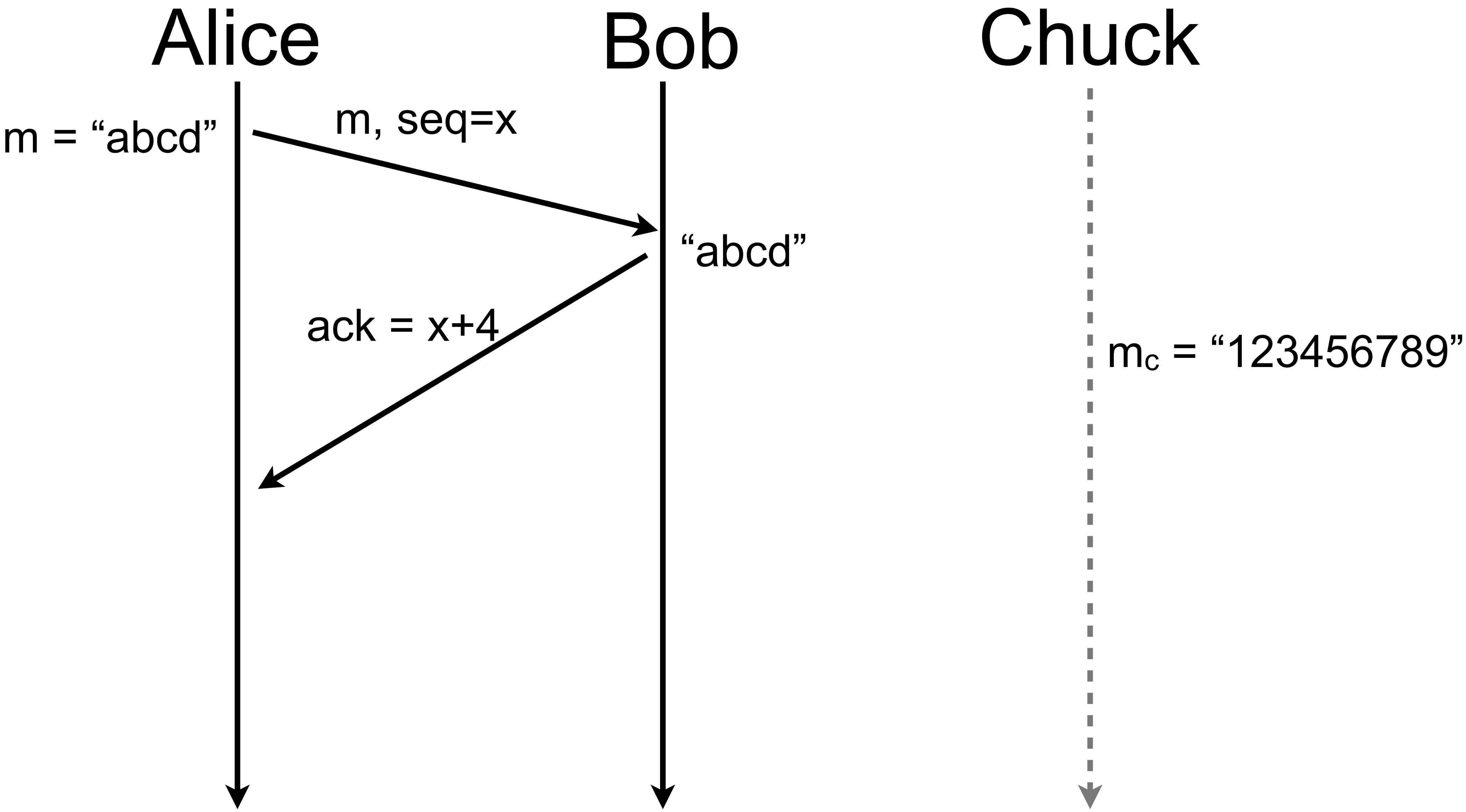
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



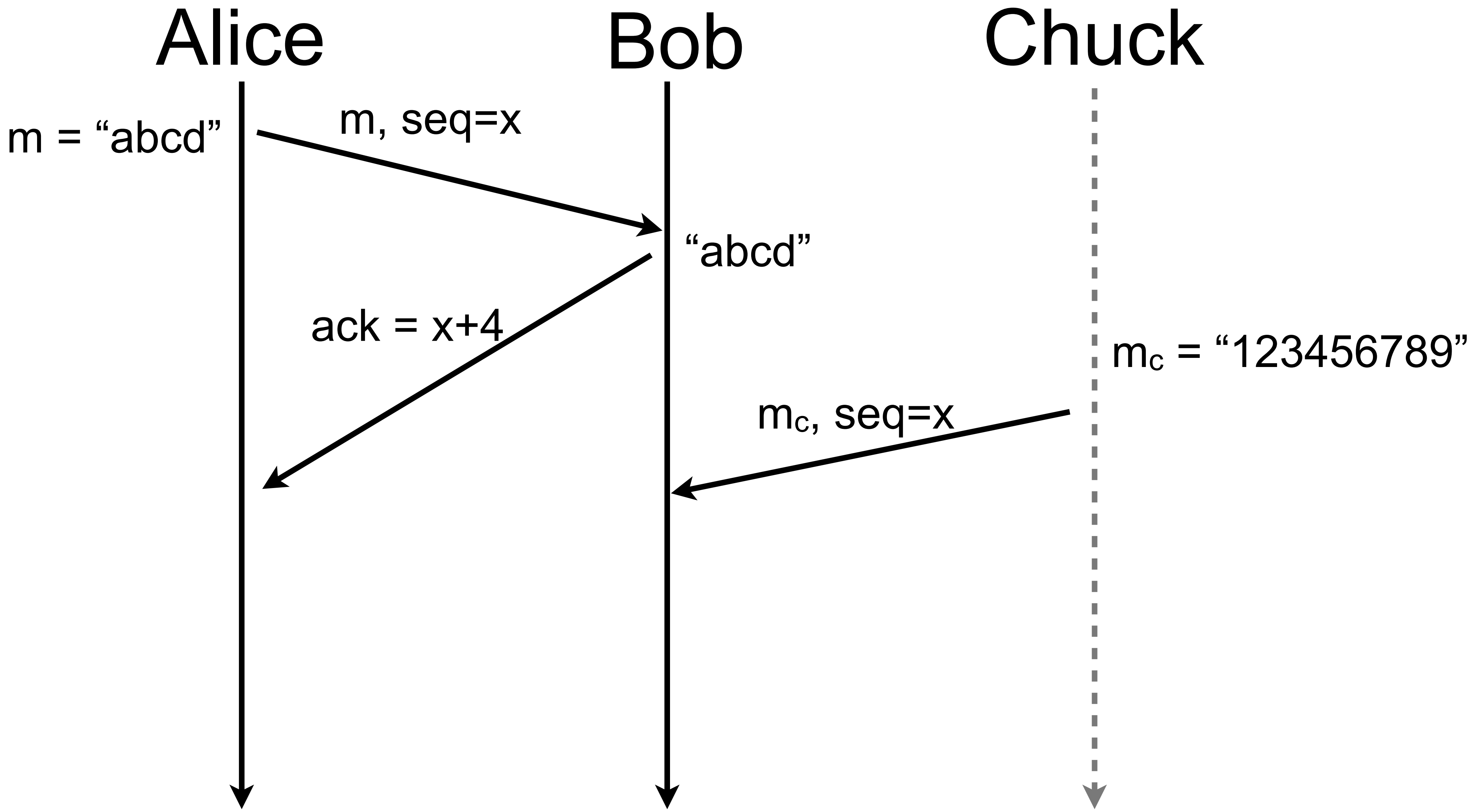
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



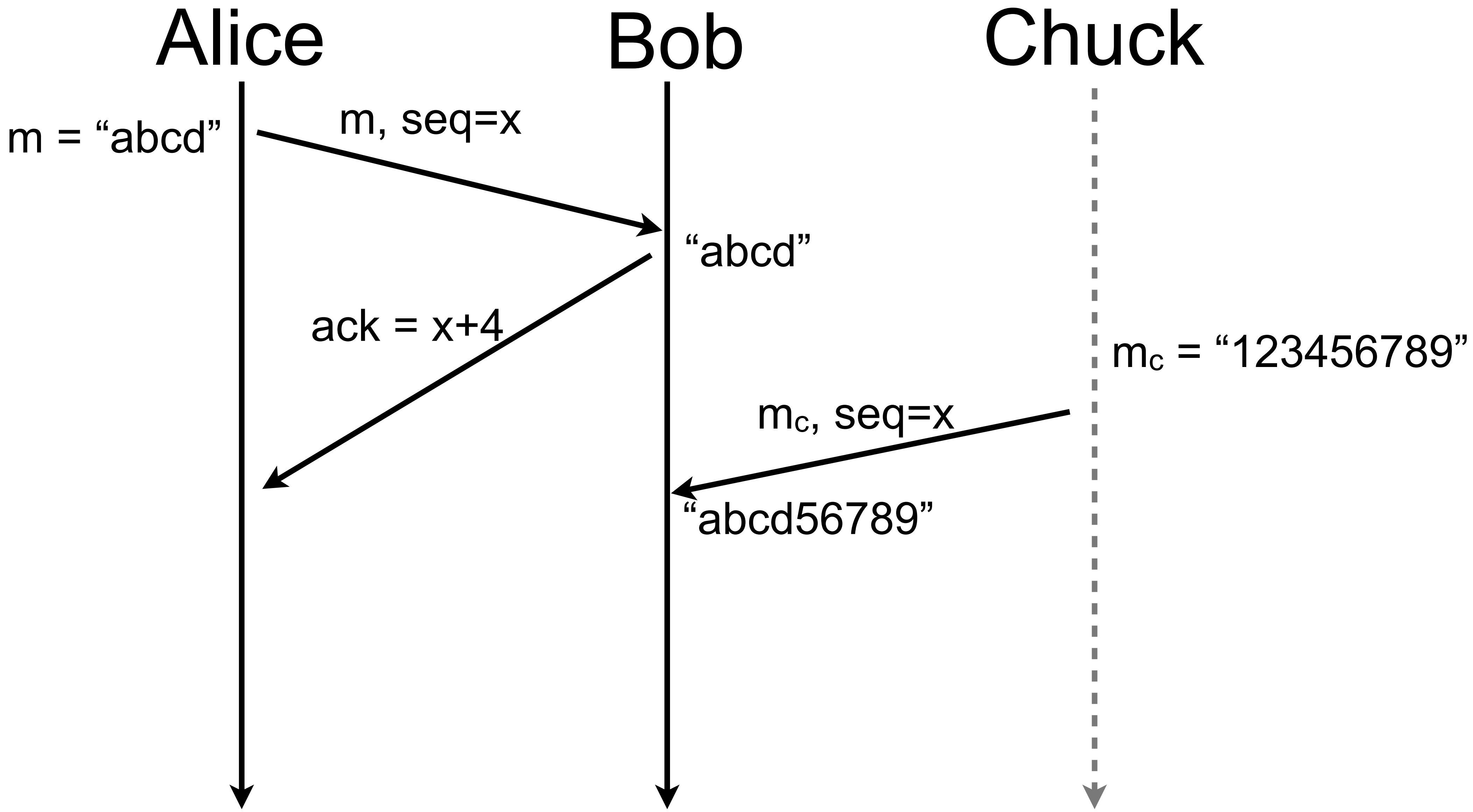
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



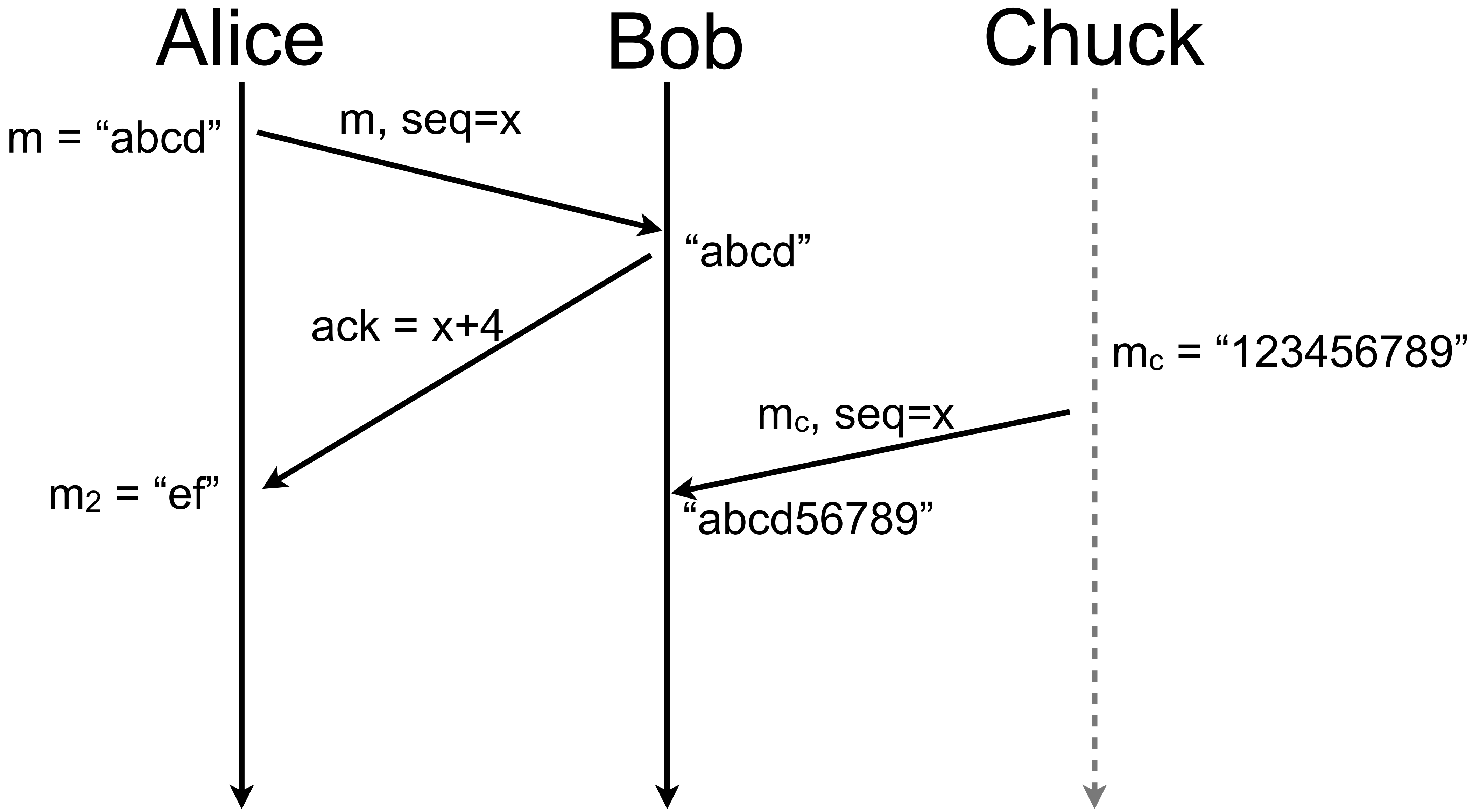
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



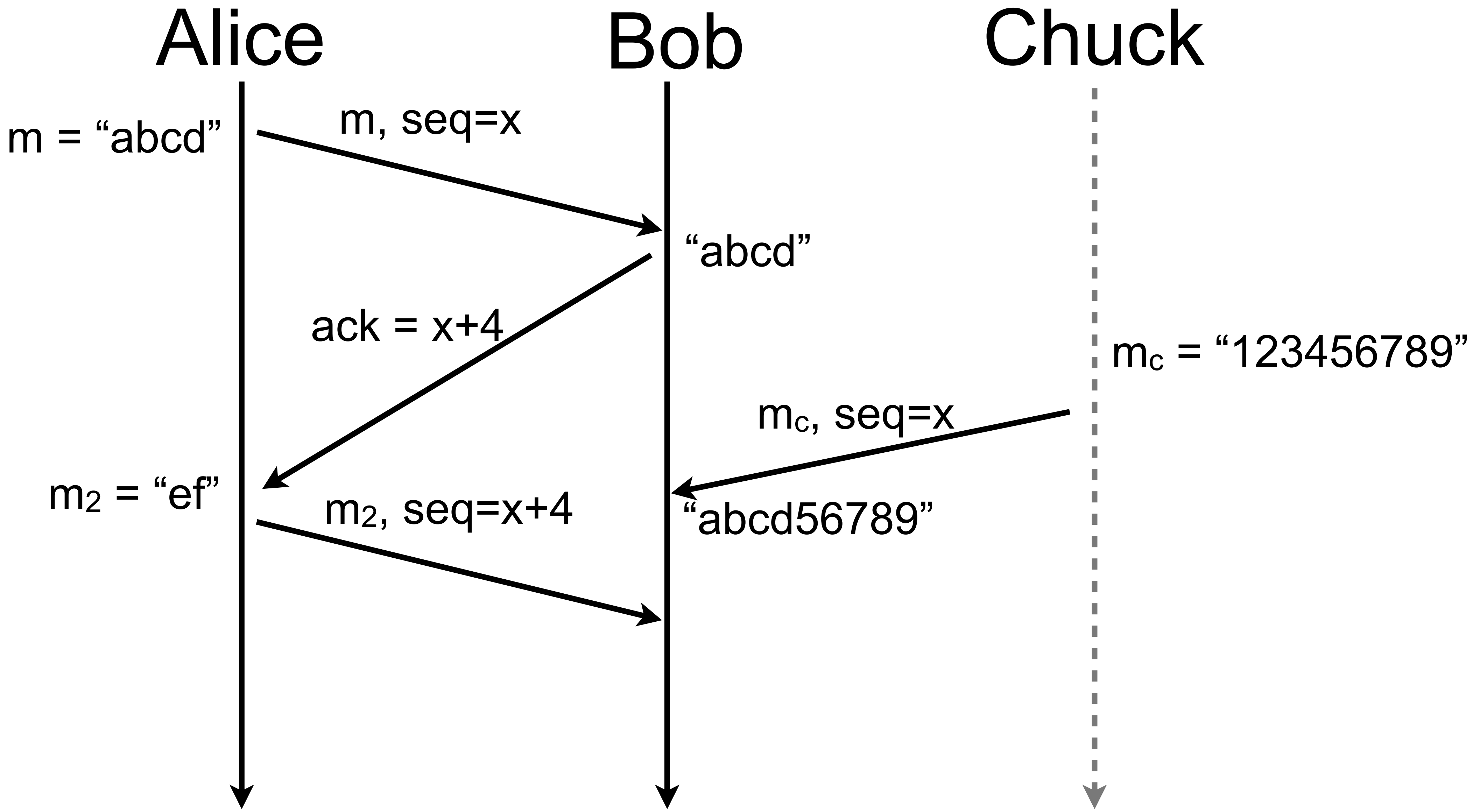
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



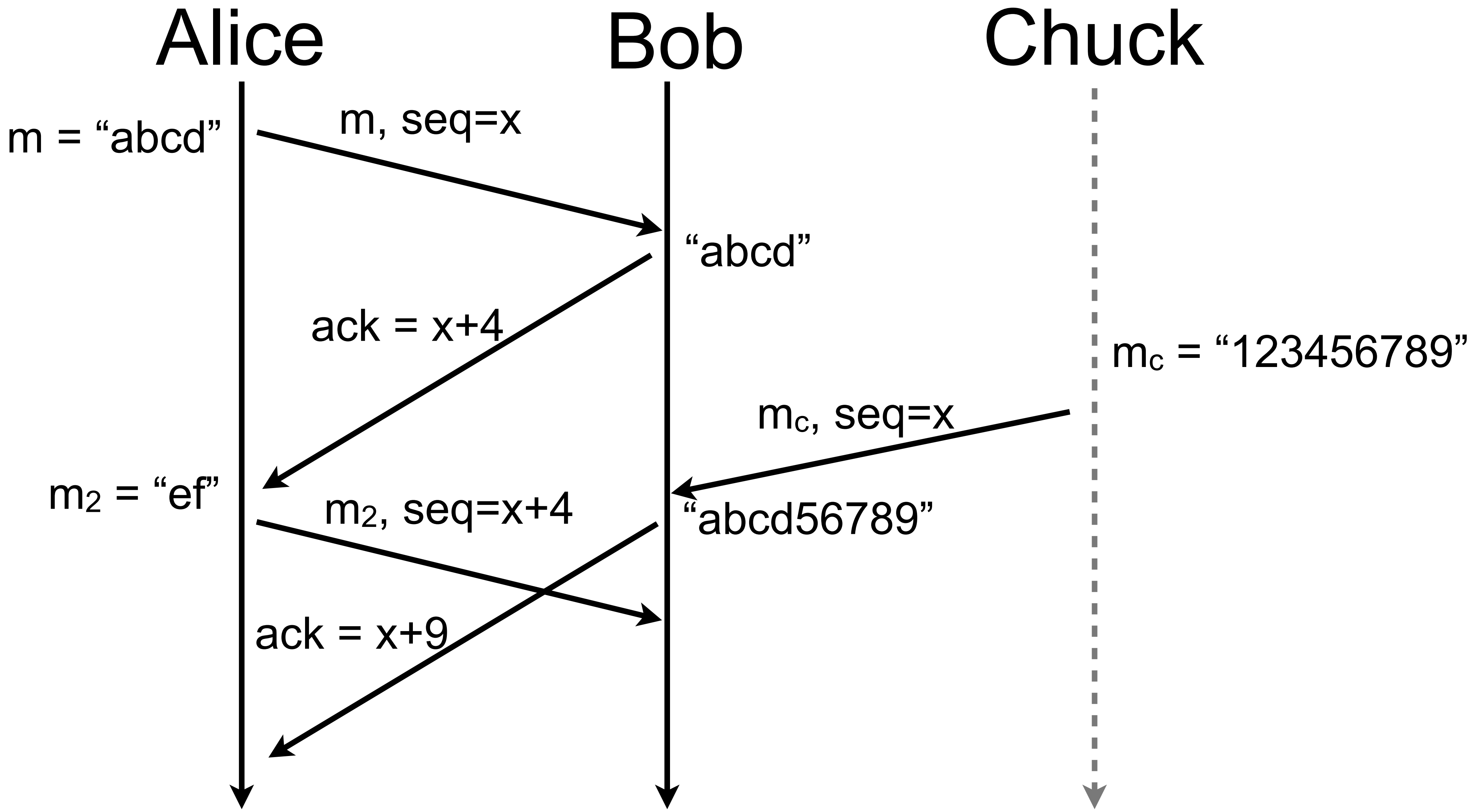
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



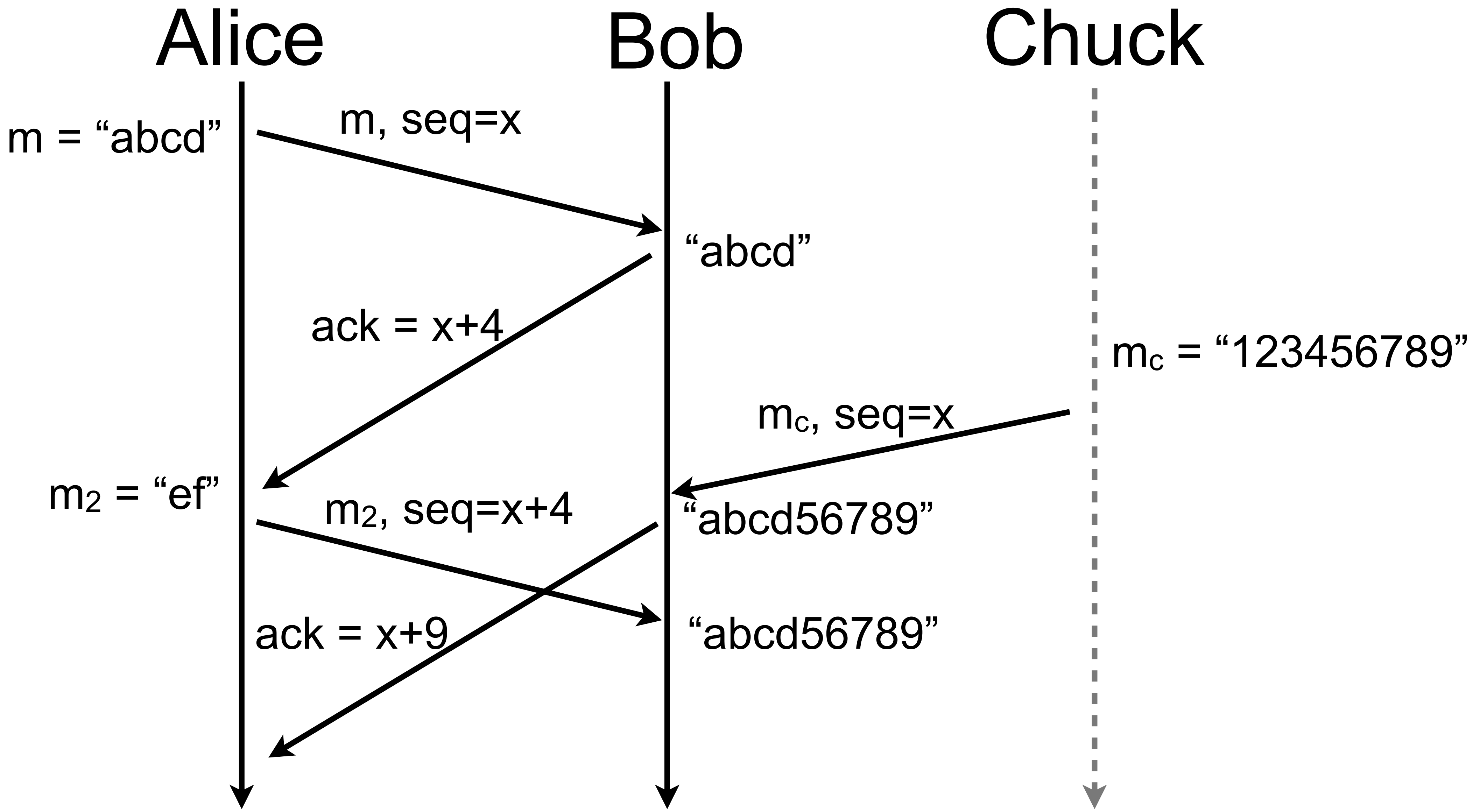
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



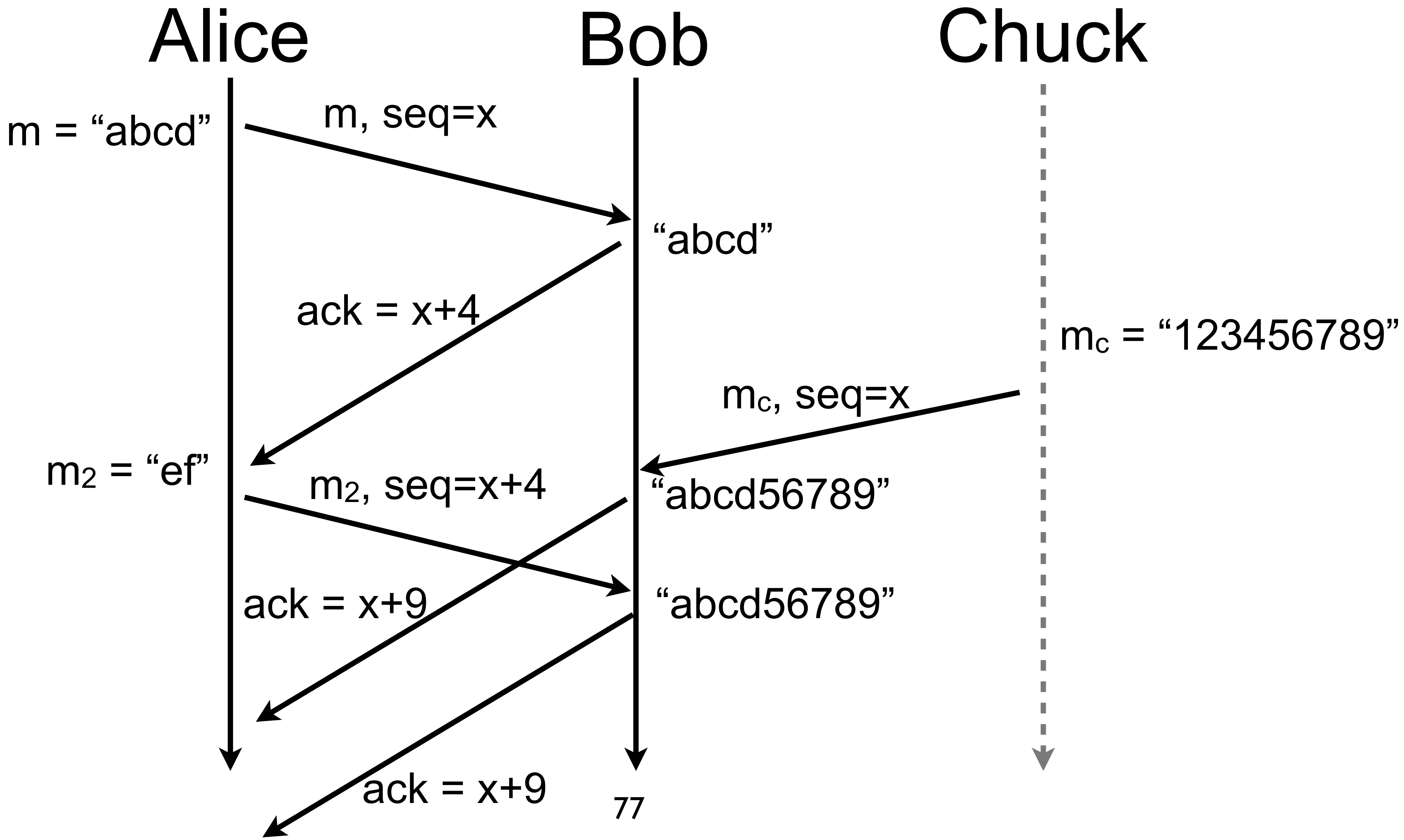
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



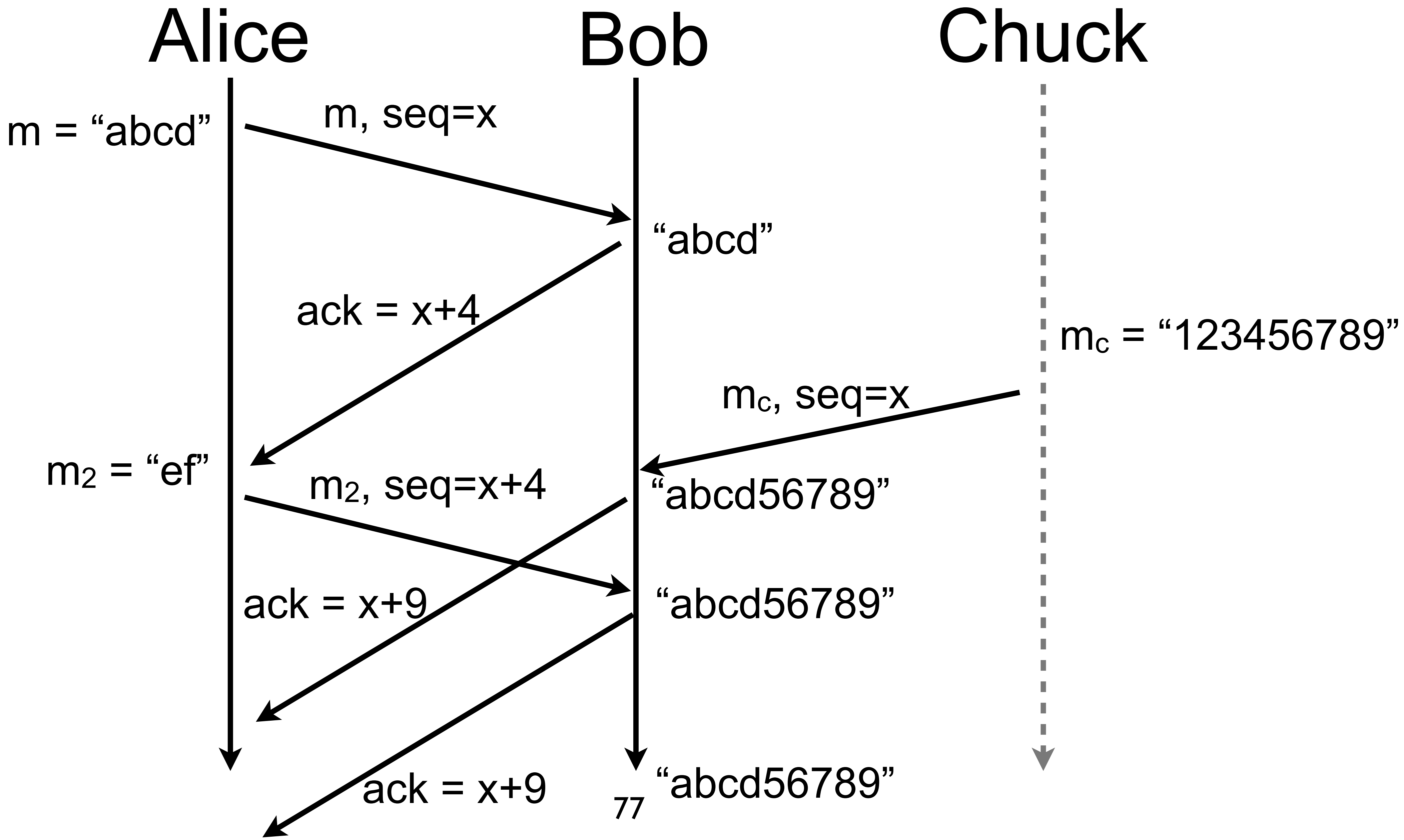
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



Problem solved?

- fill me
- fill me
- fill me

Problem solved?

- fill me
- fill me
- fill me

DoS attacks are still possible!

Denial of Services

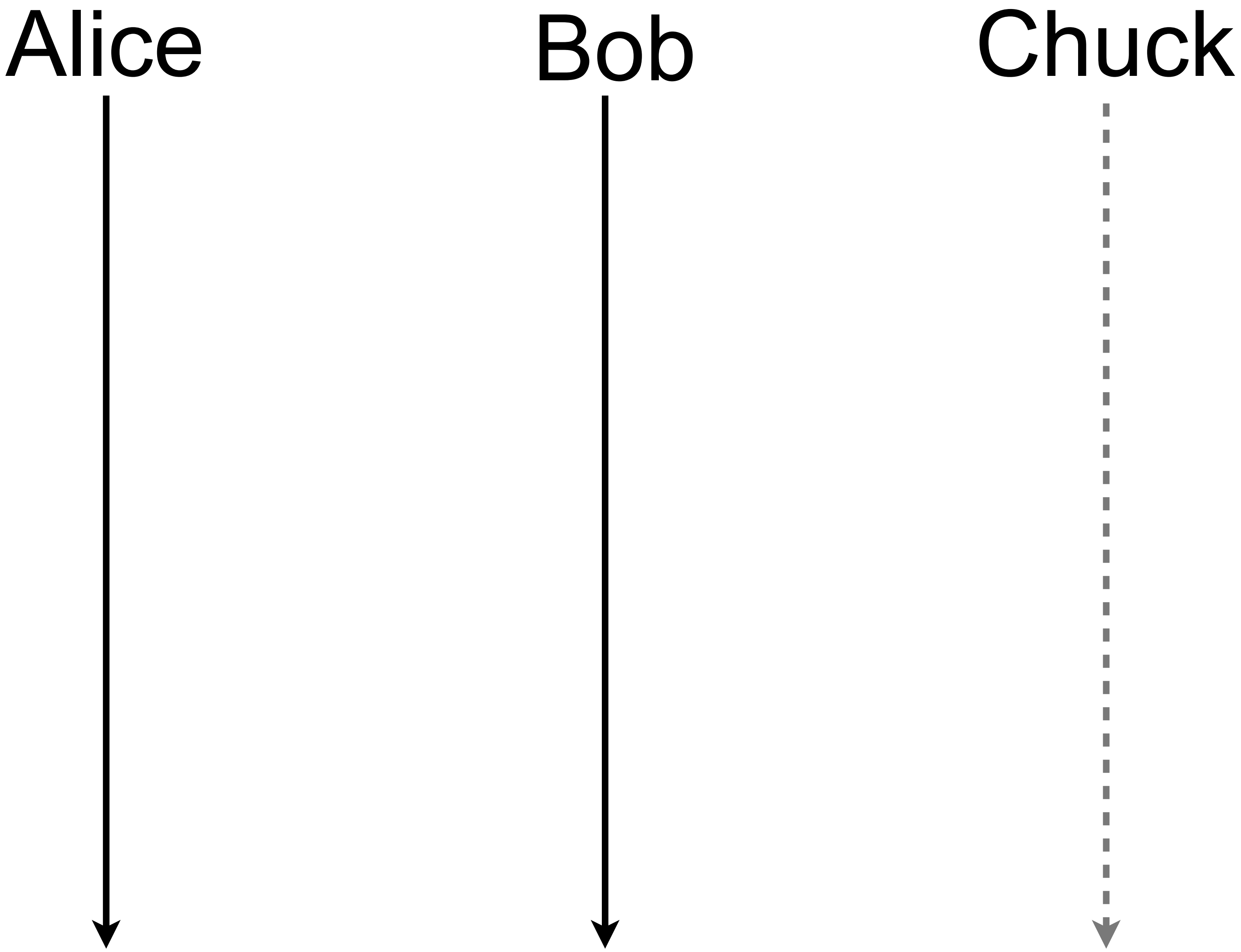
- Resources are always limited
  - e.g., processor, memory, link capacity
- The easiest way of leading a DoS is to overwhelm CPUs, memory, or links of the target
- A more complicated way is to manage an intrusion and neutralize the target
  - imagine you gain administrative access to border router of your network!

Danger of state

- Establishment and maintenance of session requires state
  - often maintained in “tables” with predefined capacity
- An attacker can saturate state tables by initiating multiple sessions
- Principle
  - require attacker to maintain state before maintaining state yourself
  - in general it is too costly for an attacker to maintain state

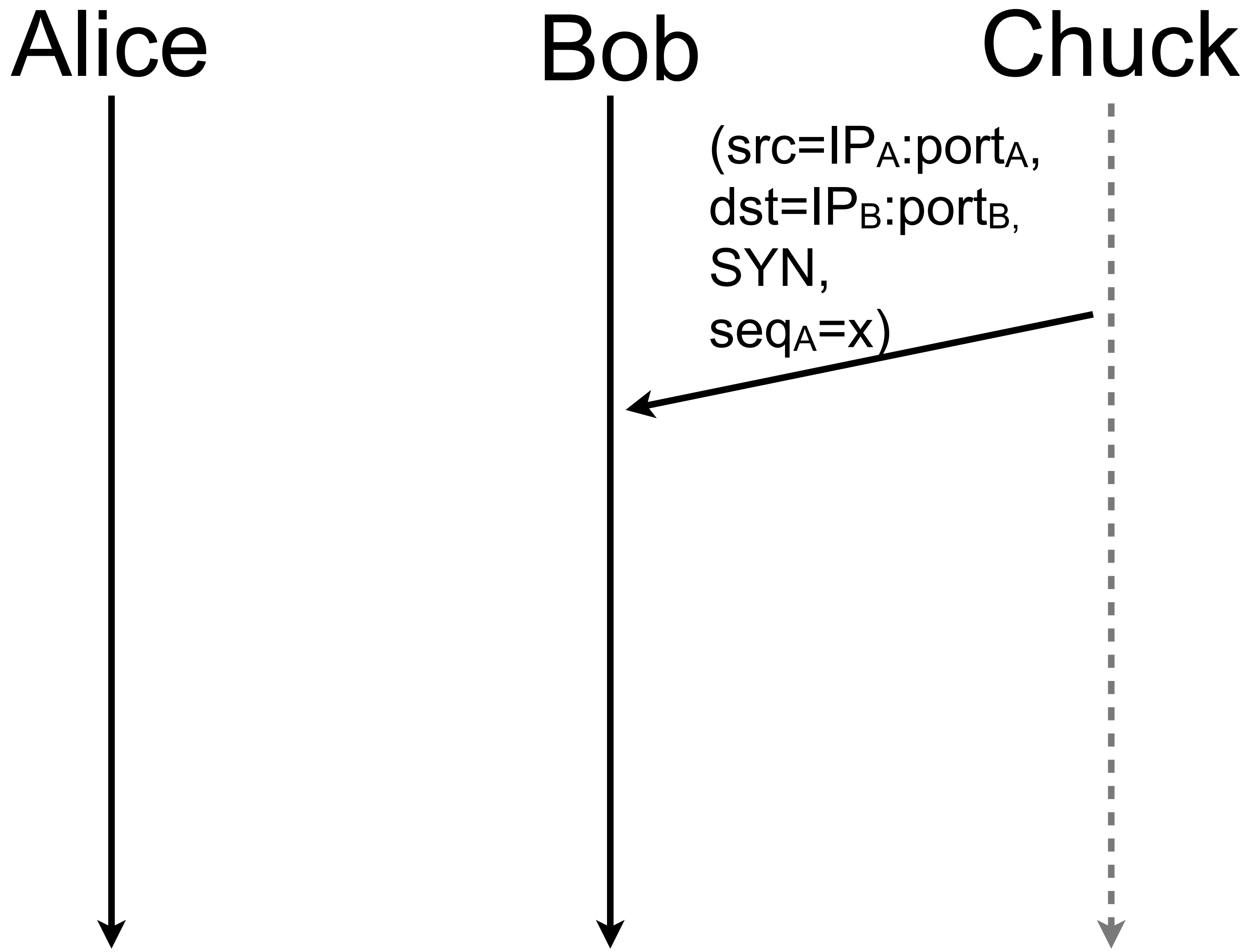
Danger of state (contd.)

- TCP relied on a state machine started upon reception of a SYN packet



Danger of state (contd.)

- TCP relied on a state machine started upon reception of a SYN packet



Alice

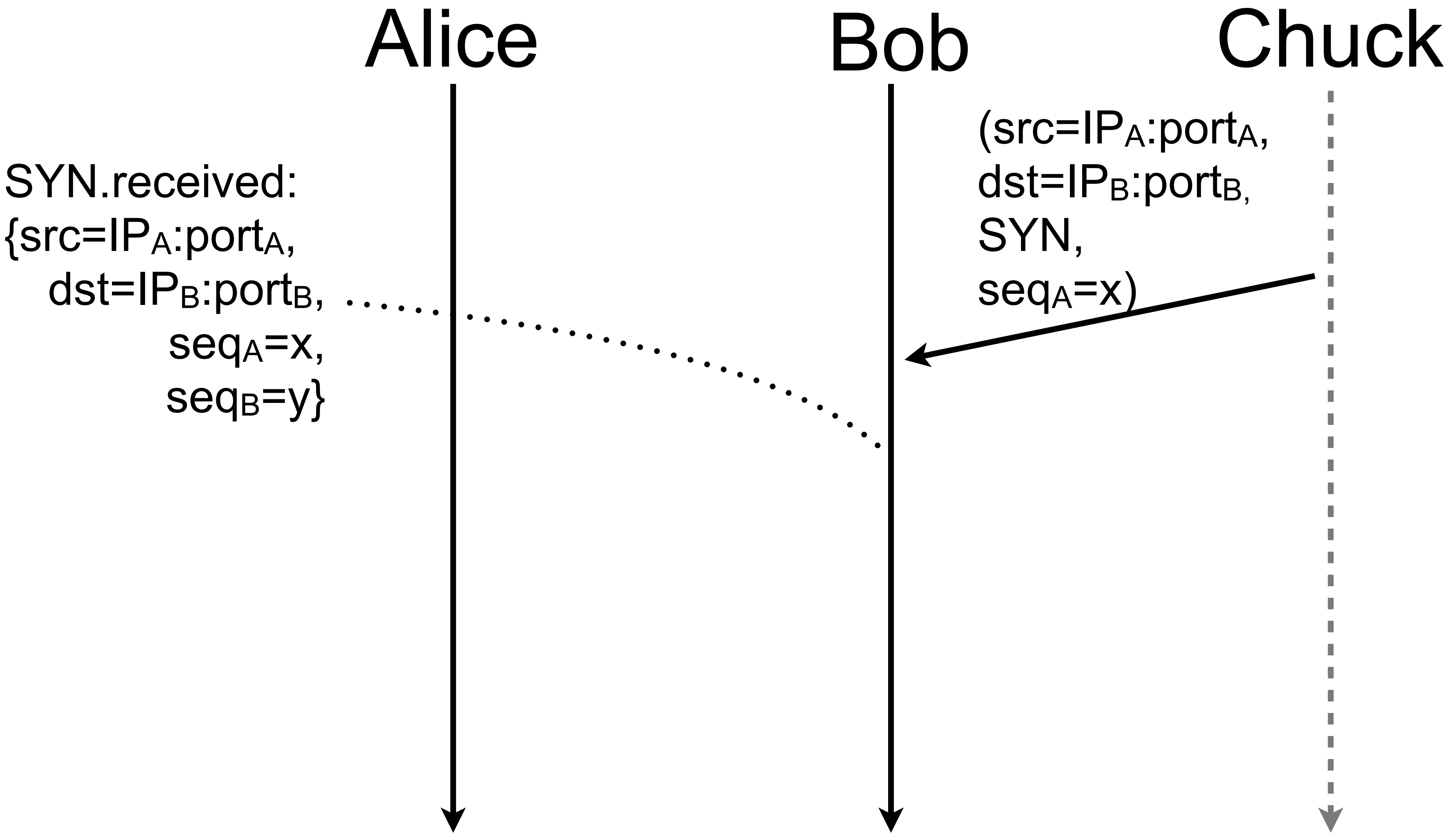
Bob

Chuck

(src=IP$_A$:port$_A$,
dst=IP$_B$:port$_{B,}$
SYN,
seq$_A$=x)

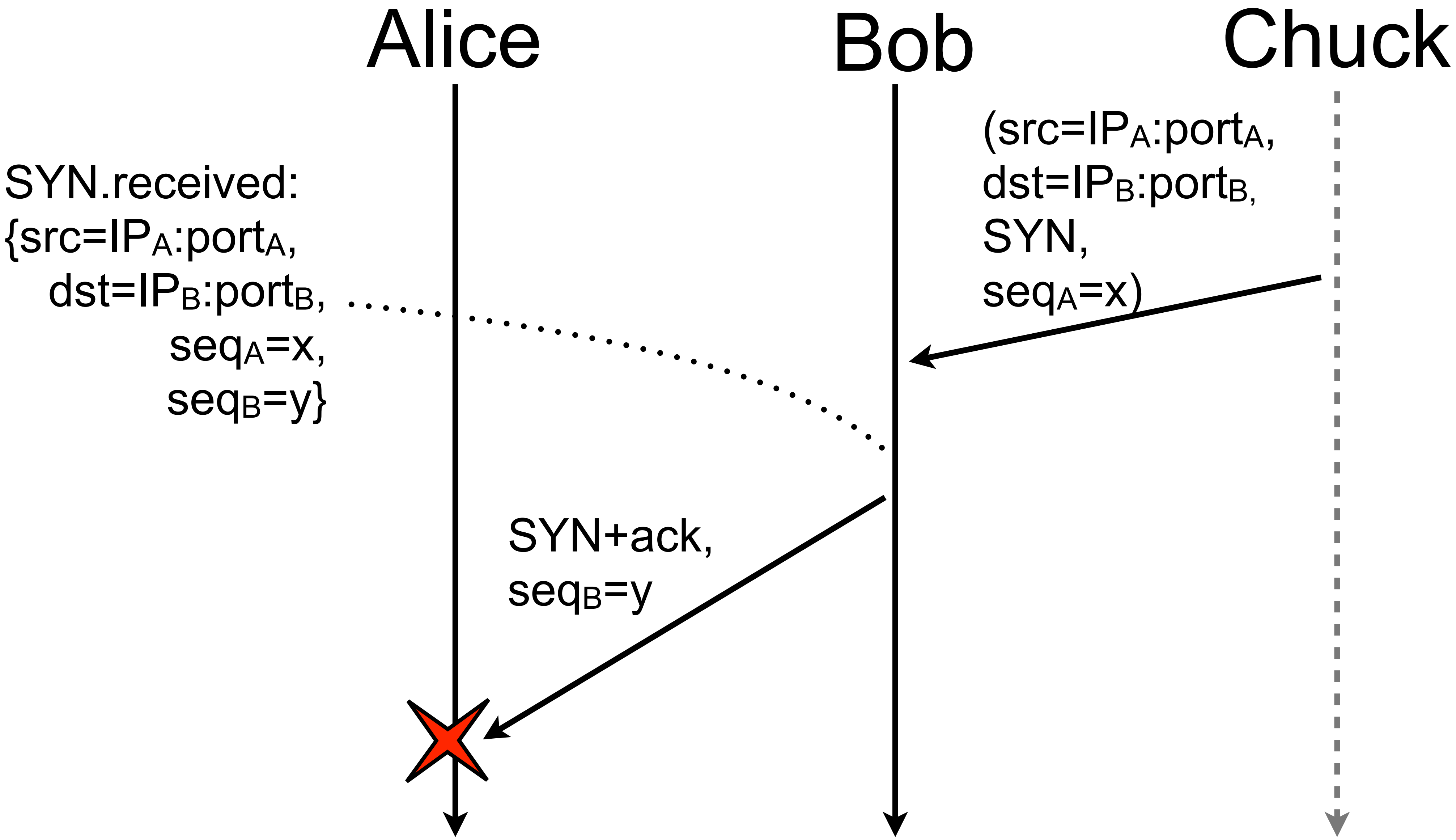
Danger of state (contd.)

- TCP relied on a state machine started upon reception of a SYN packet



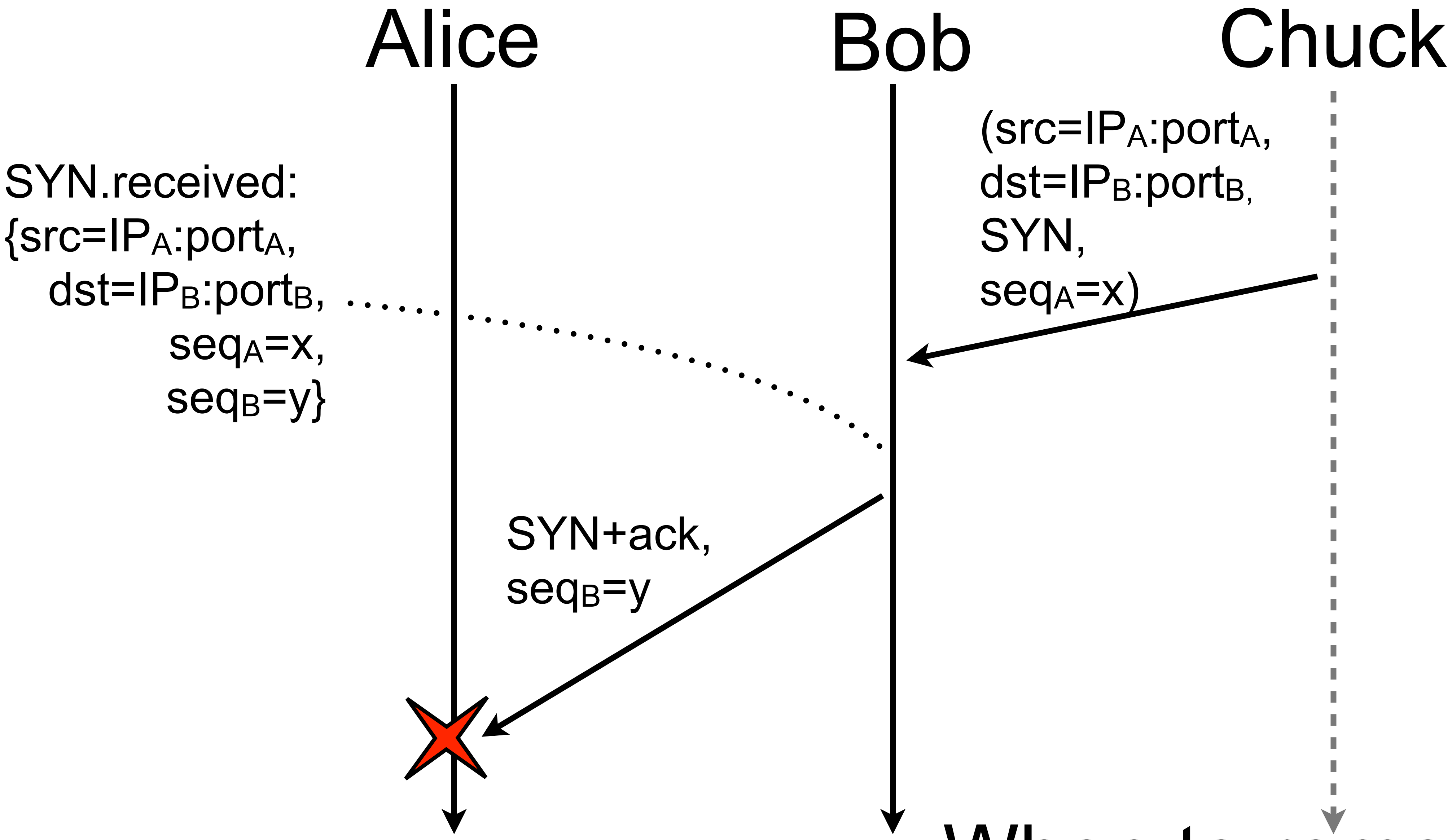
Danger of state (contd.)

- TCP relied on a state machine started upon reception of a SYN packet



Danger of state (contd.)

- TCP relied on a state machine started upon reception of a SYN packet



Alice

Bob

Chuck

SYN.received:
{src=IP_A:port_A,
dst=IP_B:port_B,
seq_A=x,
seq_B=y}

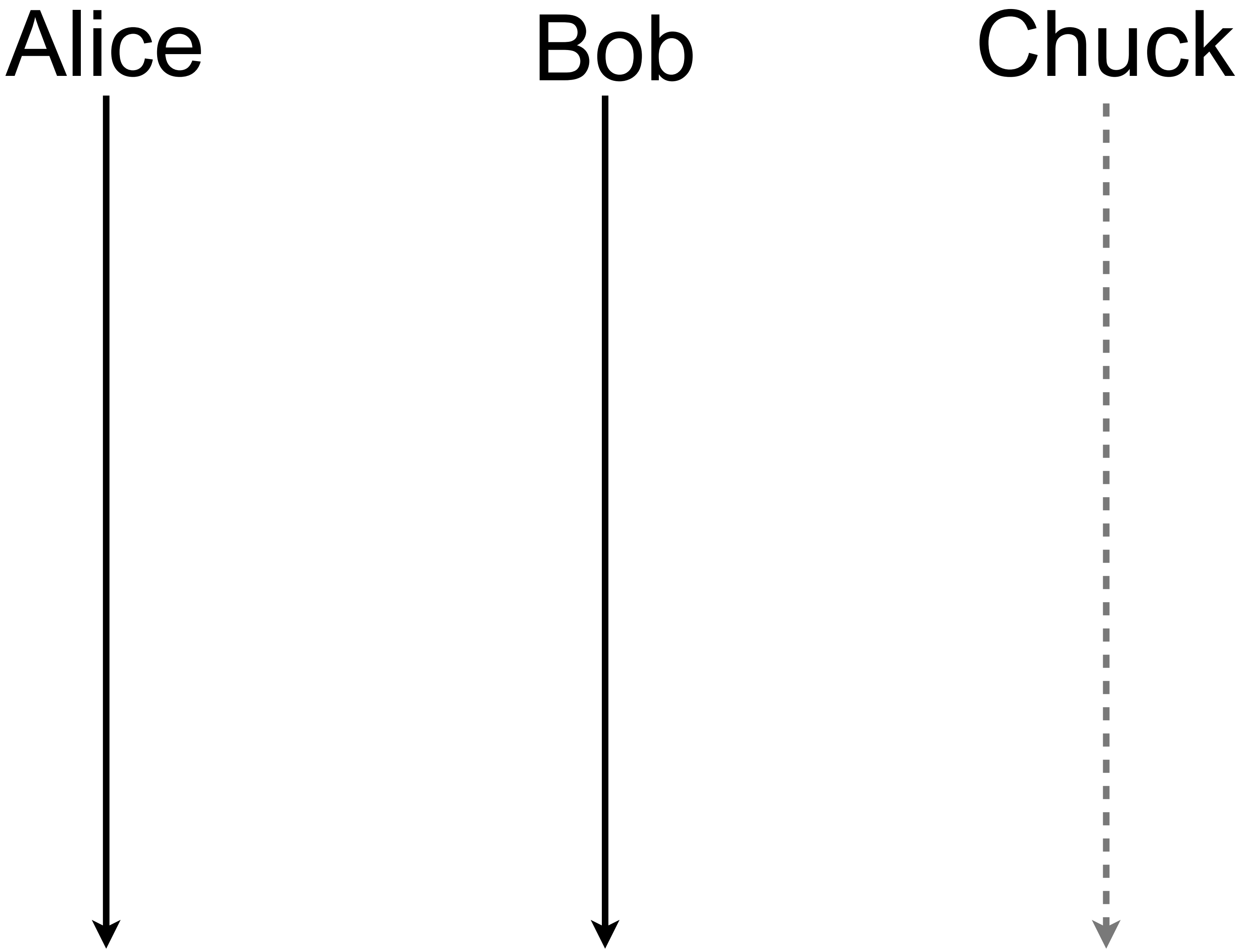
(src=IP_A:port_A,
dst=IP_B:port_B,
SYN,
seq_A=x)

SYN+ack,
seq_B=y

When to remove state?

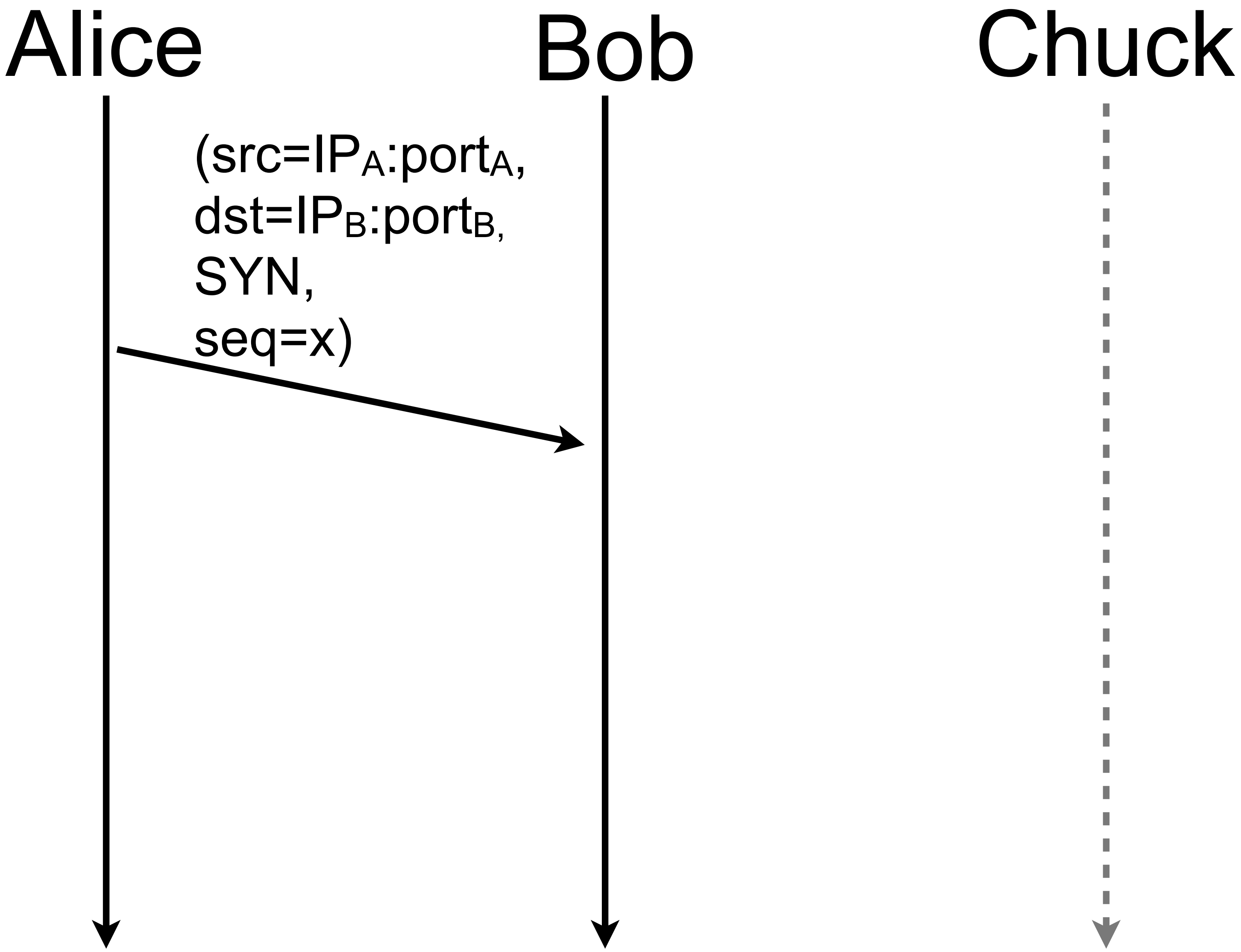
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



Alice

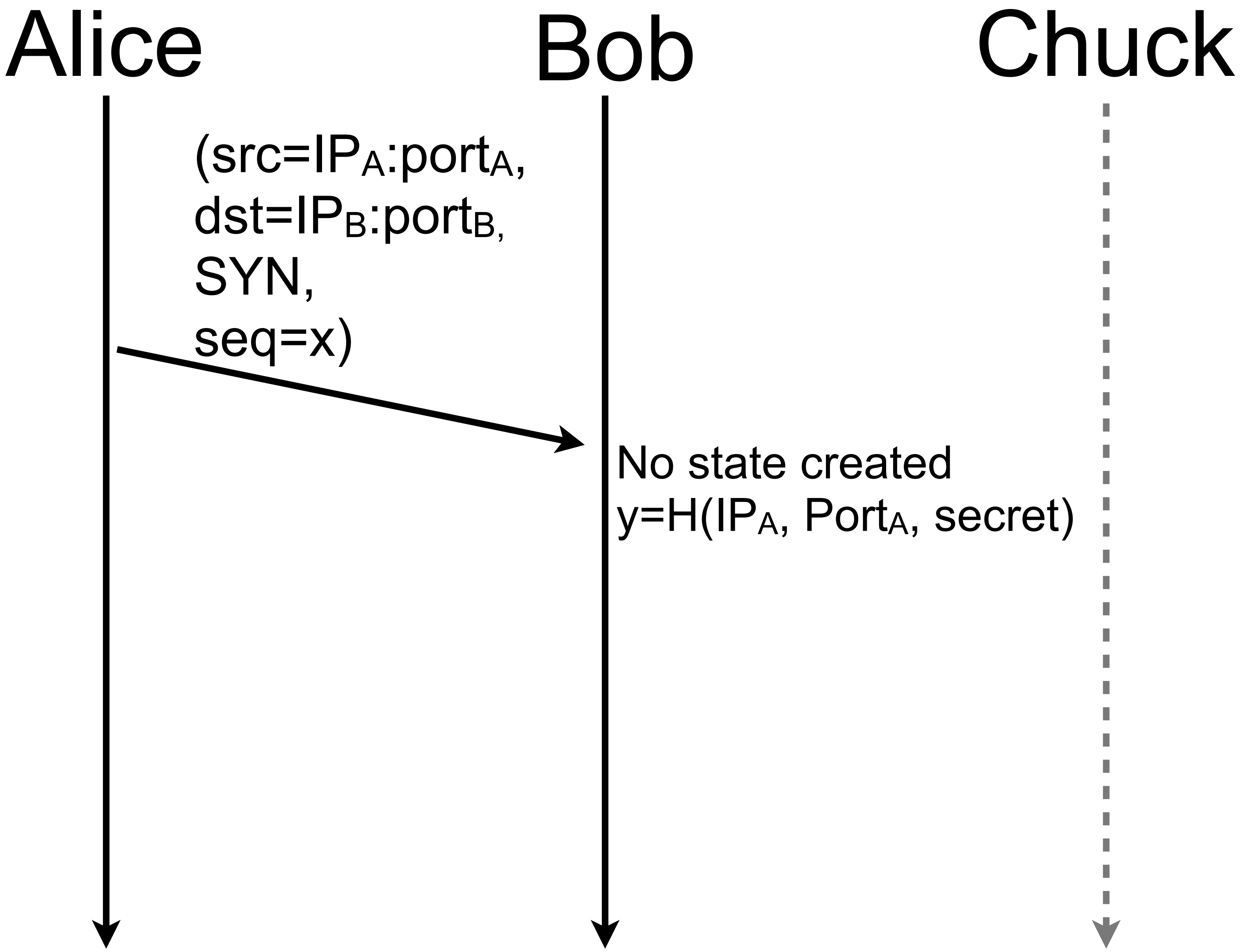
Bob

Chuck

(src=IP$_A$:port$_A$,
dst=IP$_B$:port$_B$,
SYN,
seq=x)

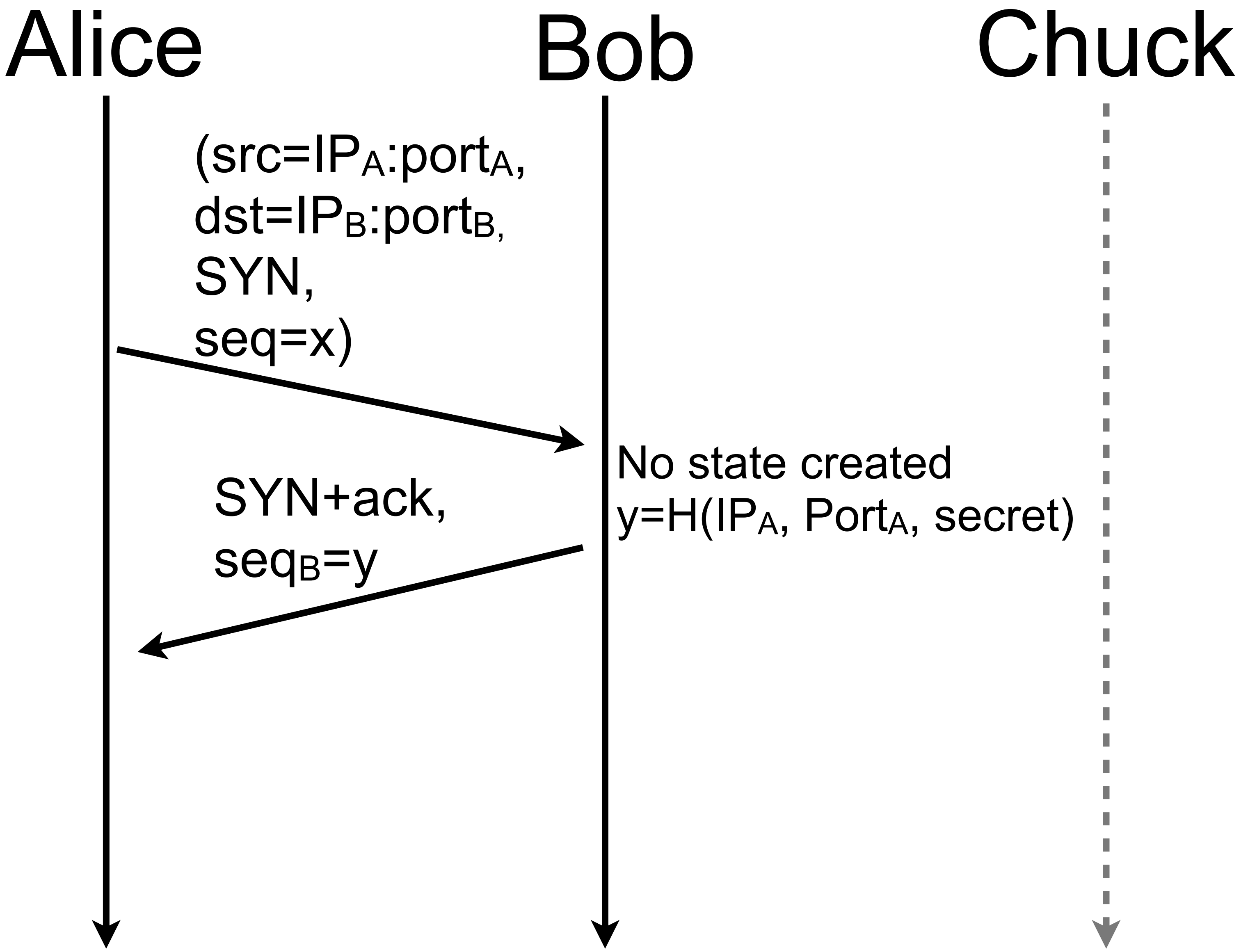
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



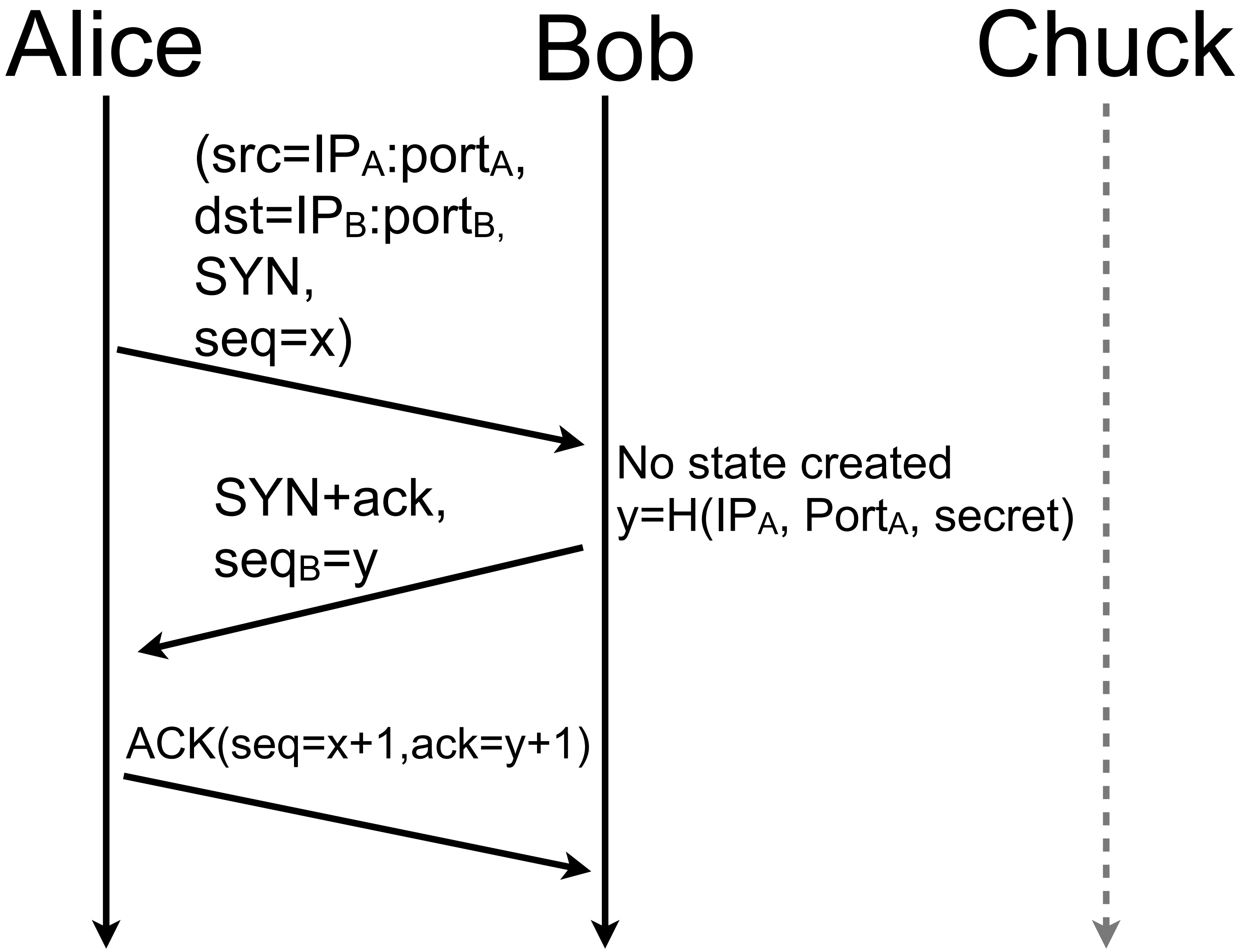
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



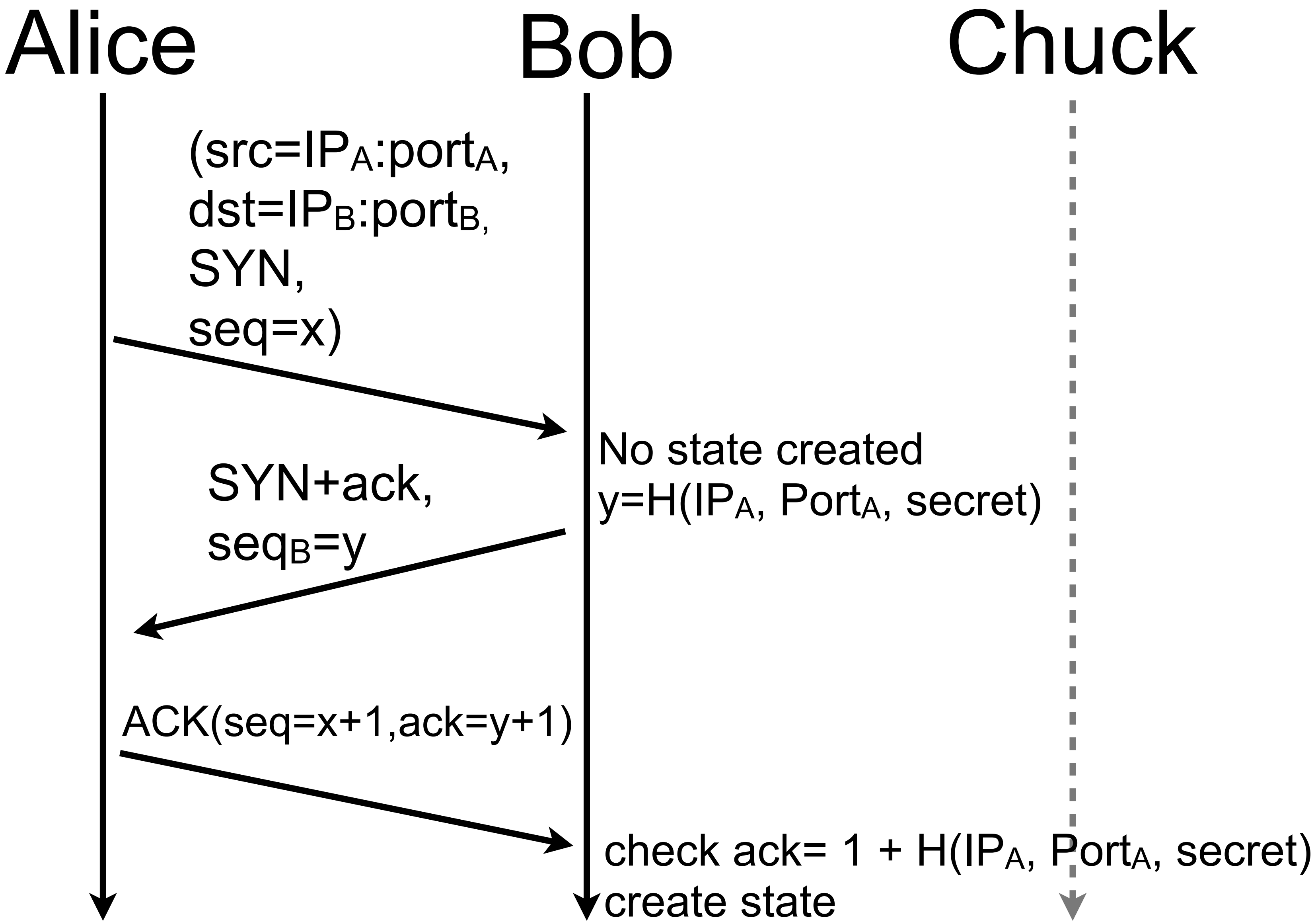
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



Alice

Bob

Chuck

(src=IP_A:$port_A$, dst=IP_B:$port_B$, SYN, seq=x)

No state created
y=H(IP_A, $Port_A$, secret)

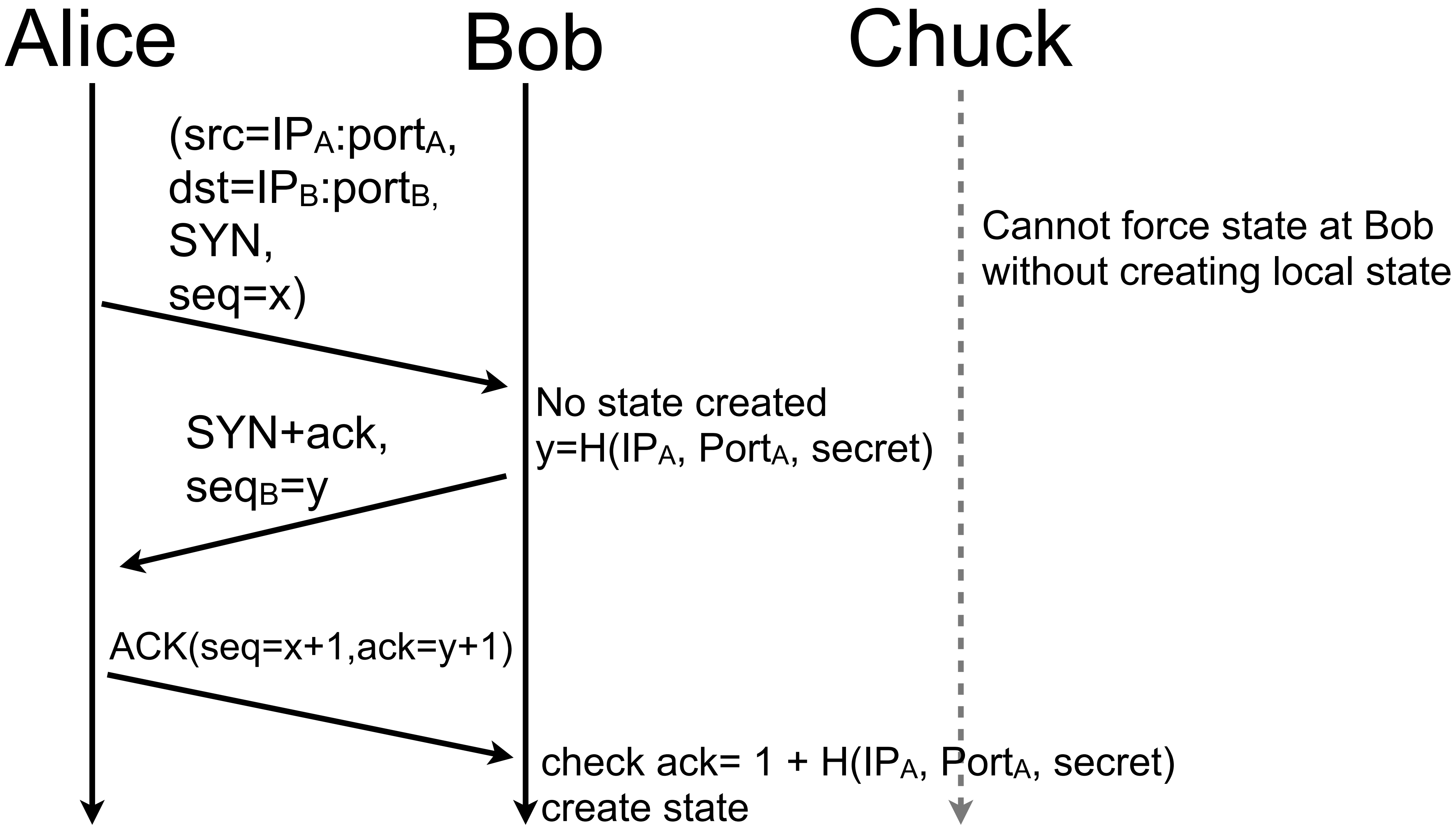
SYN+ack, seq_B=y

ACK(seq=x+1,ack=y+1)

check ack= 1 + H(IP_A, $Port_A$, secret)
create state

Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



Alice

Bob

Chuck

(src=IP_A:$port_A$,
dst=IP_B:$port_{B,}$
SYN,
seq=x)

No state created
y=H(IP_A, $Port_A$, secret)

SYN+ack,
seq_B=y

ACK(seq=x+1,ack=y+1)

check ack= 1 + H(IP_A, $Port_A$, secret)
create state

Cannot force state at Bob
without creating local state

Danger of complexity

- Protection mechanism can be complex and can require important processing power
- An attacker can overwhelm her target CPU by triggering protection mechanisms
- Principle
  - require attacker to perform more processing than yourself
  - in general an attacker does not want to have to do heavy computation

Danger of complexity (contd.)

- Hard, if not impossible, to remove processing requirements but still possible to force the attacker to succeed some challenges to get access. This technique is usually called challenge-response
  - time challenges
    - when an attack is suspected, force the attacker to wait or slow down but the DoS protection can lead to a DoS
      - e.g., rate limiting
  - mathematical challenges
    - ask the initiator to solve a mathematical challenge that is hard to compute but easy to check, this might negatively impact legitimate clients
    - e.g., Bob asks Alice to find a J such that the K lowest order bits of H((N,J)) are zeros. N is a nonce and K sets the complexity of the puzzle, both parameters are decided by Bob [RFC5201]
  - human processing challenge
    - some services are reserved for users and don’t want to be accessed by bots
    - ask Alice to succeed a challenge that is simple for a human but hard for a computer
      - e.g., CAPTCHA

Danger of complexity (contd.)

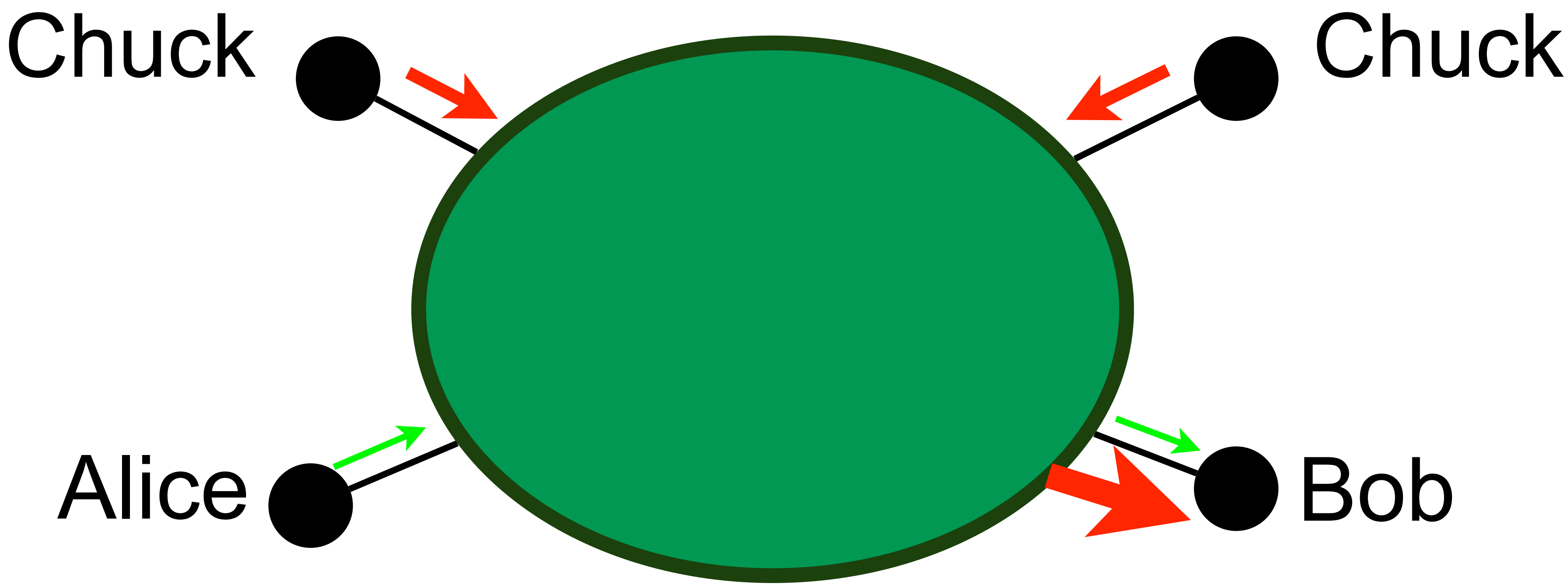
- Hard, if not impossible, to remove processing requirements but still possible to force the attacker to succeed some challenges to get access. This technique is usually called challenge-response
  - time challenges
    - when an attack is suspected, force the attacker to wait or slow down but the DoS protection can lead to a DoS
      - e.g., rate limiting
  - mathematical challenges
    - ask the initiator to solve a mathematical challenge that is hard to compute but easy to check, this might negatively impact legitimate clients
    - e.g., Bob asks Alice to find a J such that the K lowest order bits of H((N,J)) are zeros. N is a nonce and K sets the complexity of the puzzle, both parameters are decided by Bob [RFC5201]
  - human processing challenge
    - some services are reserved for users and don’t want to be accessed by bots
    - ask Alice to succeed a challenge that is simple for a human but hard for a computer
      - e.g., CAPTCHA

Link overloading

- Messages are sent to Bob by traversing links
- If an attacker can send packets at a high enough rate, she can saturate links toward Bob and make him unavailable
- Unfortunately, Bob cannot make anything to block packet before they reach him
- Principle
  - tweak the network to not suffer too much of such attacks

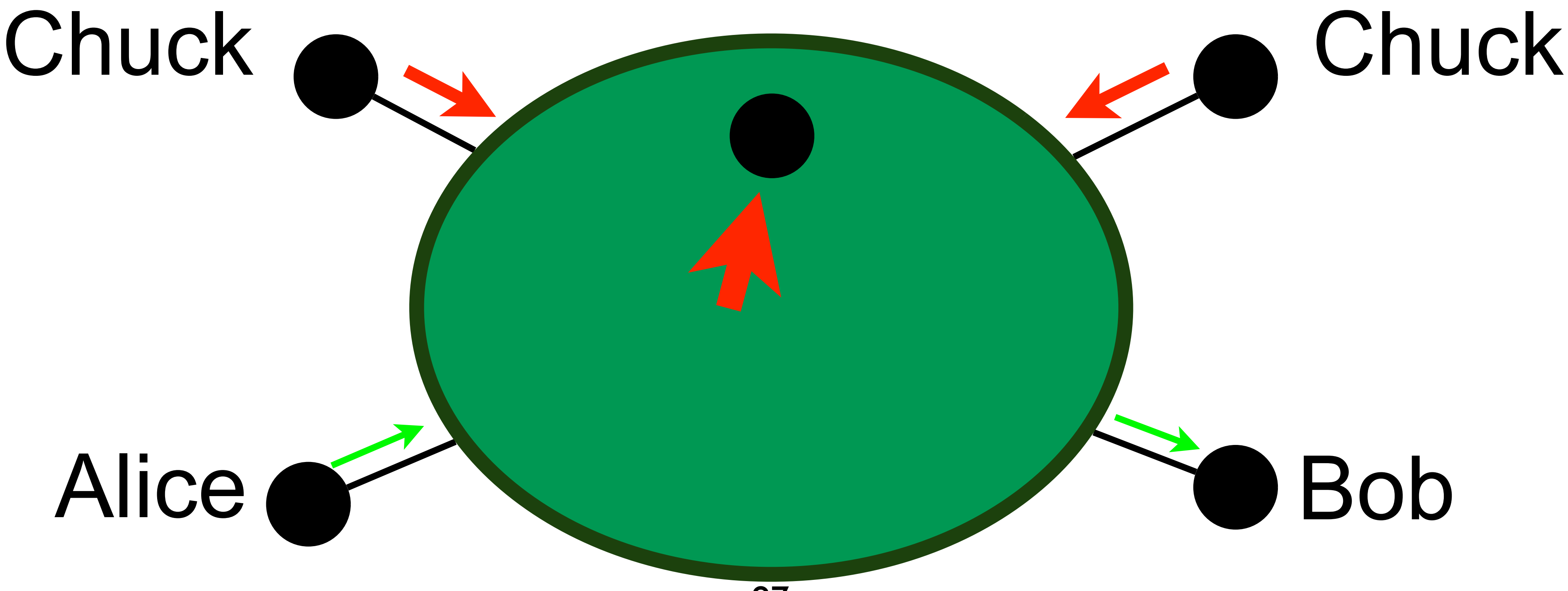
Link overloading (contd.)

- Example of Distributed Denial of Service (DDoS) attack



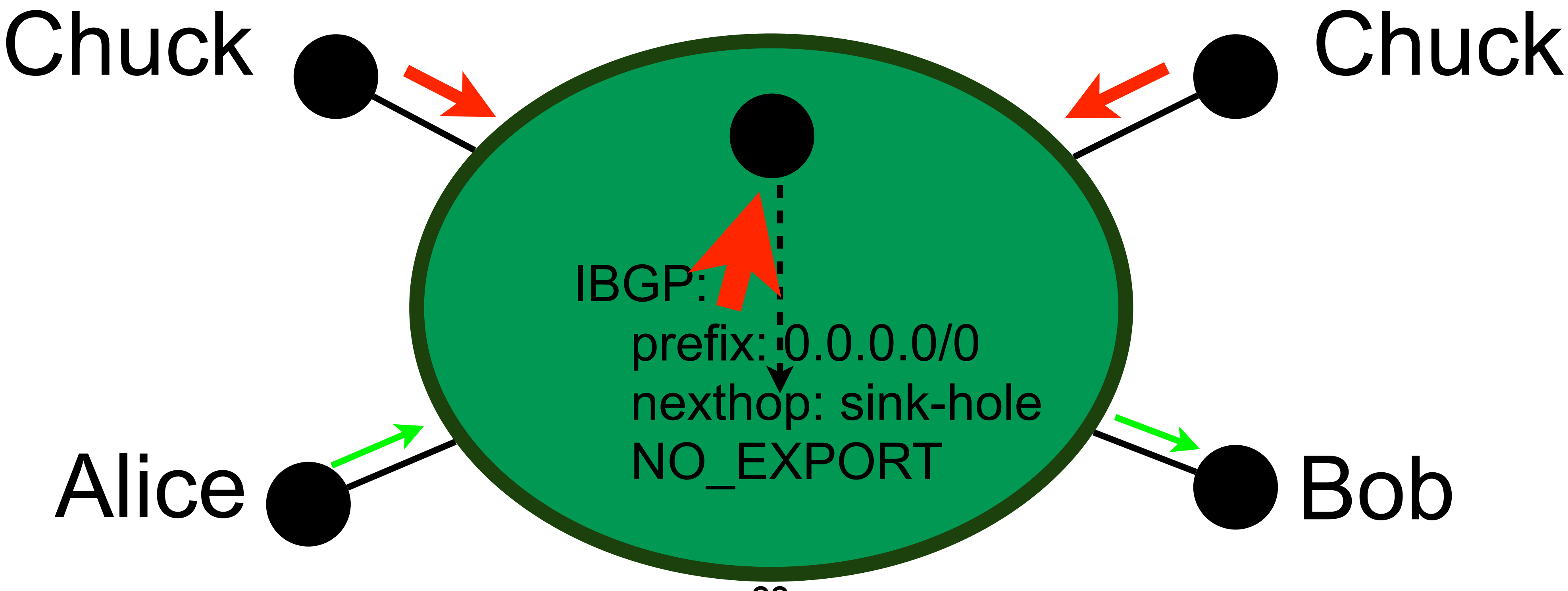
Link overloading (contd.)

- Attacks are often to random destinations or with random sources
  - backscatter traffic to a sink-hole that can receive a lot of traffic attack without impacting the network



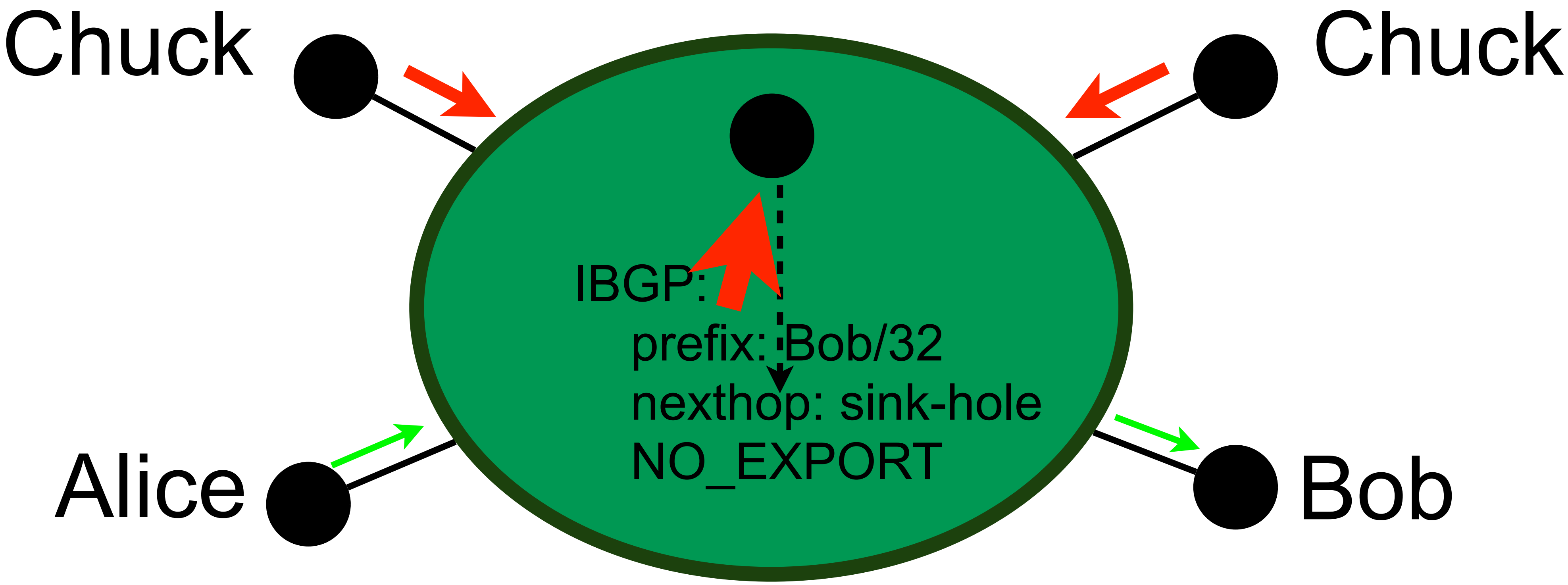
Link overloading (contd.)

- Use the sink-hole to attract bizarre packets



Link overloading (contd.)

- Use the sink-hole to protect the target



Link overloading (contd.)

- A first parade is to filter illicit traffic before it can harm the target
  - e.g., firewall, access lists
- A set of rules is specified a priori, if the traffic does not match the rules, it is discarded
  - always block everything but what is acceptable

Link overloading (contd.)

- Filtering based on origin
  - useful to avoid spoofing
    - e.g., block any packet which source address does not belong to the customer cone of a BGP neighbor
  - does not work so well as it depends on every network between the origin and the target
- Filtering based on traffic pattern
  - analyze the traffic and if it deviates from what is normal, drop it
    - e.g., drop malformed packets, rate limit a source if it sends too much SYN packets, ignore mails from well known SPAM servers, block any flow initiated by the outside if there is no server in the network

Network Intrusion Detection System (NIDS)

- An NIDS aims at discovering non-legitimate operations
- The NIDS analyses the traffic to detect abnormal patterns
- Upon anomaly detection, the NIDS triggers an alert with a report on the anomaly
- NOC follows procedures upon detection

Network Intrusion Detection System (contd.)

- Signature based detection
  - a database of abnormal behavior is maintained to construct a signature for each attack
  - if the traffic corresponds to a signature in the database, trigger an alarm
  - risk of false negative (0-day attack)
  - e.g., Snort, Bro, antivirus
- Outlier detection
  - the anomaly detector learns what is the normal behavior of the network
  - went an outlier is detected, an alarm is triggered
  - risk of false positive and false negative
  - e.g., cluster analysis, time series analysis, spectral analysis

- if antivirus(self) == BAD:
  - skip
- else:
  - I am bad

Problem solved?

- fill me
- fill me
- fill me

Problem solved?

- fill me
- fill me
- fill me

Relay attacks are still possible!

Relay attack

- In a relay attack, Chuck does not contact Alice directly but goes via Bob
- If the traffic from Bob to Alice is bigger than the traffic from Chuck to Bob, the attack is called amplification attack
- As for DoS, hard to protect correctly against relay attacks
  - use filters (e.g., deactivate ICMP)
  - authentication of the source
    - but correct spoofing protection that doesn't open a relay attack door is very hard to deploy in practice as it requires messages in both directions between parties

What did we miss?

What did we miss?

- To terminate the session!
  - with the same care as the opening of the session
  - this is often neglected

Perfect Forward Secrecy

- With perfect forward secrecy (PFS), Eve cannot decrypt messages sent between Alice and Bob
  - even if she captures every message
  - even if she breaks into Alice and Bob after the communication to steal their secrets (e.g., private keys)

Perfect Forward Secrecy (contd.)

- PFS is provided using ephemeral keys
  - the ephemeral key is generated and used only during the session
  - the session key is not stored after the communication
  - the session key is independent of stored information (e.g., good PRNG)
  - for long sessions, change the session key regularly

Perfect Forward Secrecy (contd.)

1. Initiate the communication between Alice and Bob
   - authenticity proven with public/private key pairs
2. Alice and Bob agree on a secret K
   - use Diffie-Hellman
     - authenticate DH messages with public/private key pairs
3. Encrypt/Decrypt messages with symmetric cryptography using K as the key
   - no need to sign as it is encrypted
   - be sure a nonce is used to avoid replay
4. If session is too long, back to 2.
5. Close the session correctly and be sure K is not stored anywhere

Blockchains

Why?

- Traditional security mechanisms rely on the notion of trust
  - who to be the trusted party (e.g., Trent)
  - concentration of power

Why?

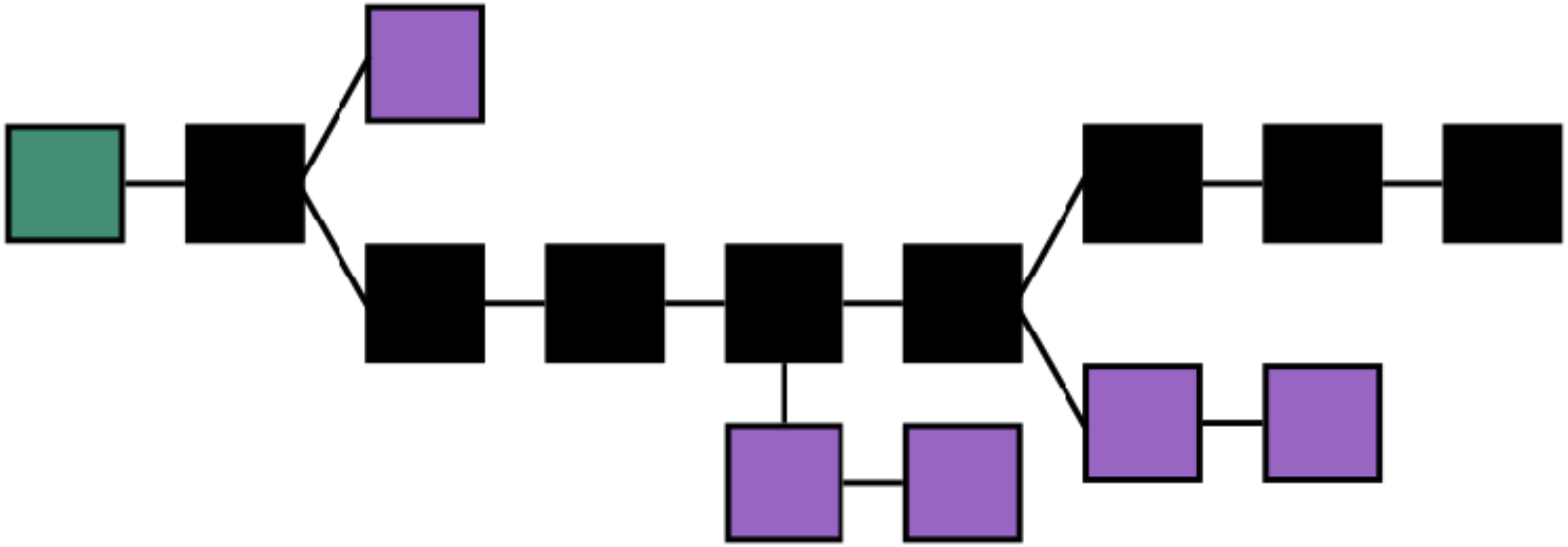
- Traditional security mechanisms rely on the notion of trust

Shift to cryptographic proof instead of trust

  - concentration of power

Definition

- *“A blockchain is a continuously growing list of records, called blocks, which are linked and secured using cryptography.”*[1]



[1] Blockchain, https://en.wikipedia.org/wiki/Blockchain, 11th Nov. 2017

First proposed with bitcoin

- Proposed for making Bitcoin transactions while avoiding double spending
  - Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." (2008): 28.
- Now blockchains go beyond transactions

First proposed with bitcoin

- Proposed for making Bitcoin transactions while avoiding double spending

In this presentation we use bitcoin as an example of blockchain

system." (2008): 28.

- Now blockchains go beyond transactions

Before the bitcoin: trust

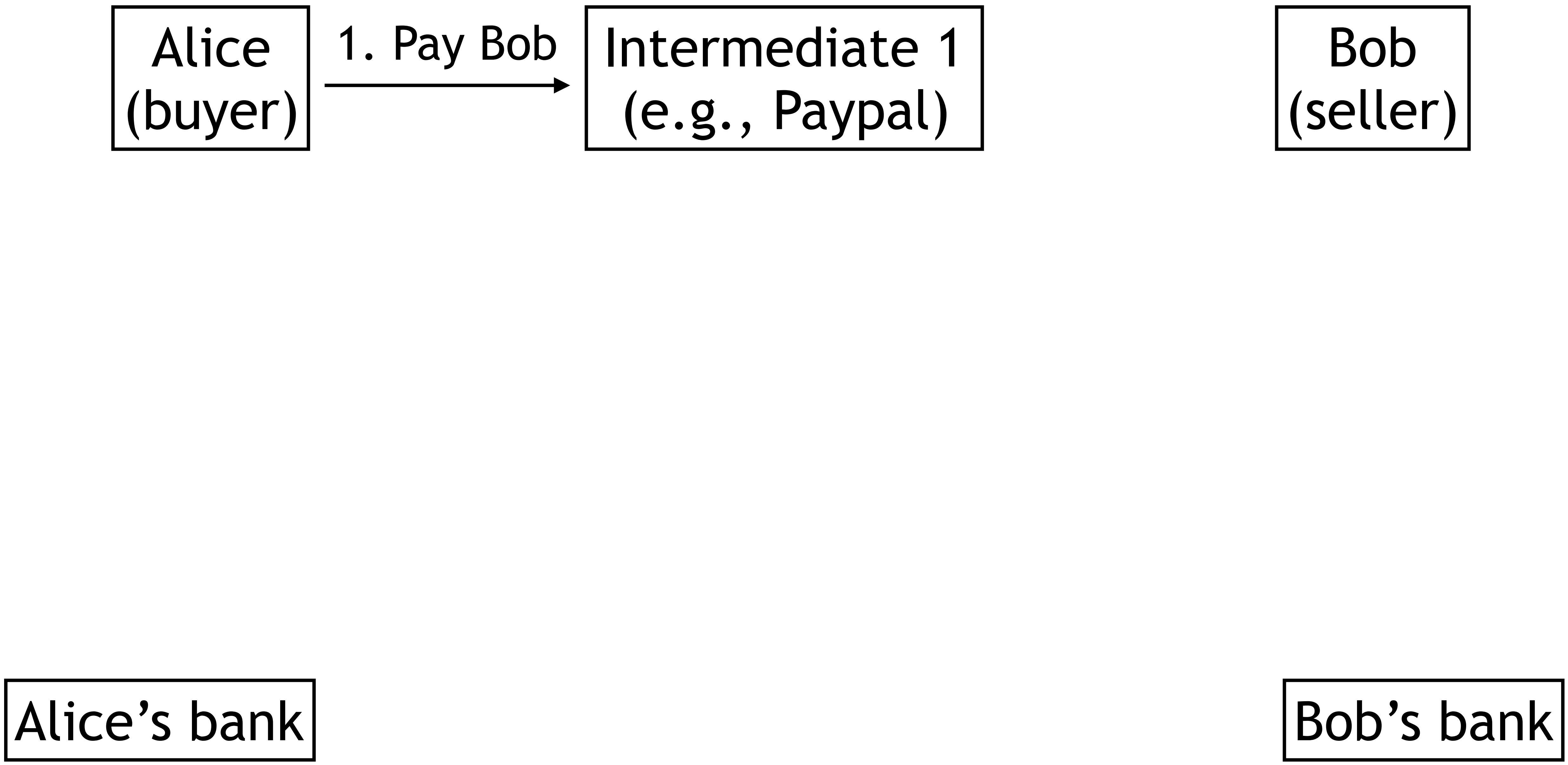
Alice (buyer)

Bob (seller)

Alice's bank

Bob's bank

Before the bitcoin: trust



Alice (buyer)

1. Pay Bob

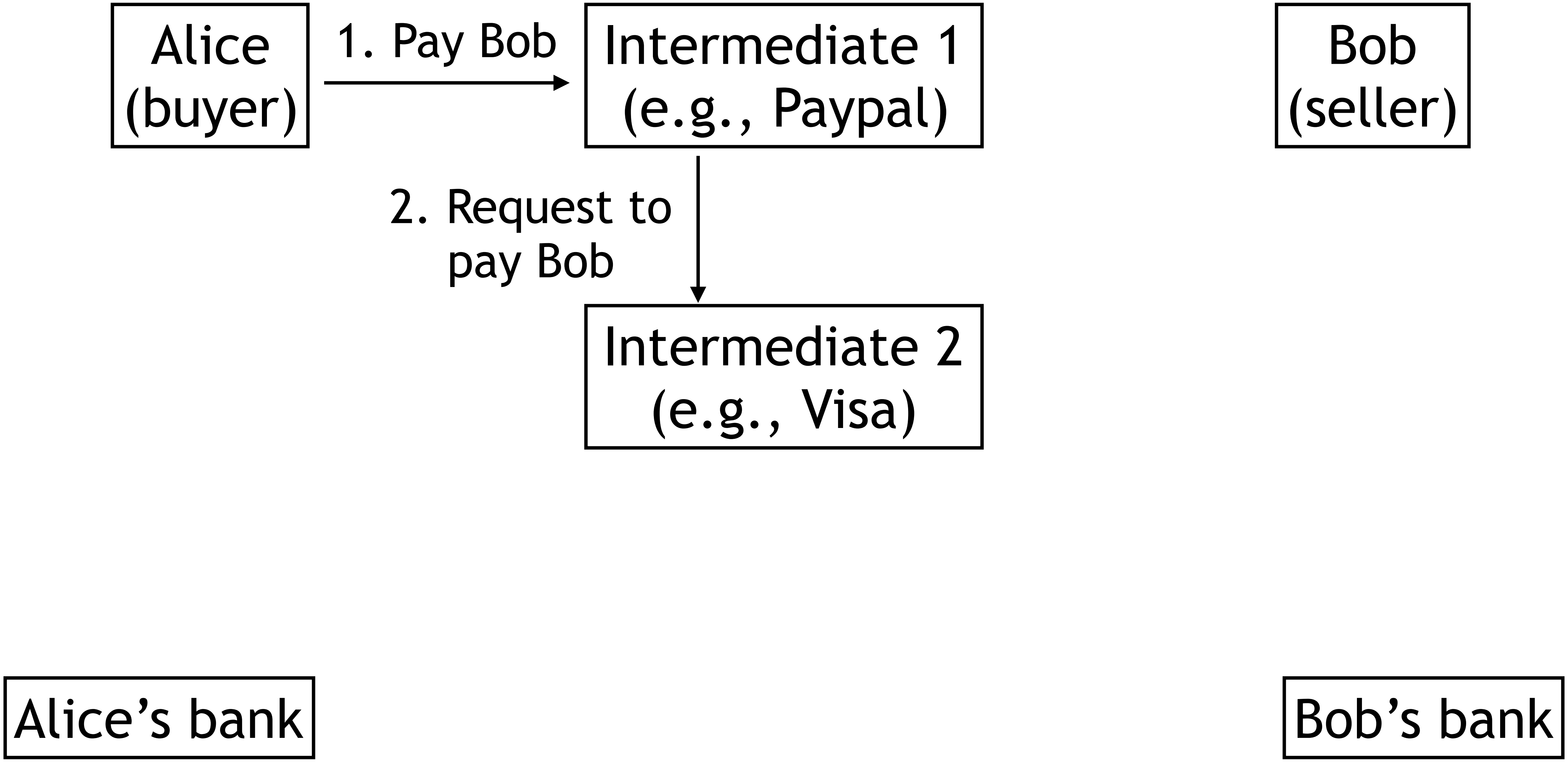
Intermediate 1 (e.g., Paypal)

Bob (seller)

Alice’s bank

Bob’s bank

Before the bitcoin: trust



Alice (buyer)

1. Pay Bob

Intermediate 1 (e.g., Paypal)

2. Request to pay Bob

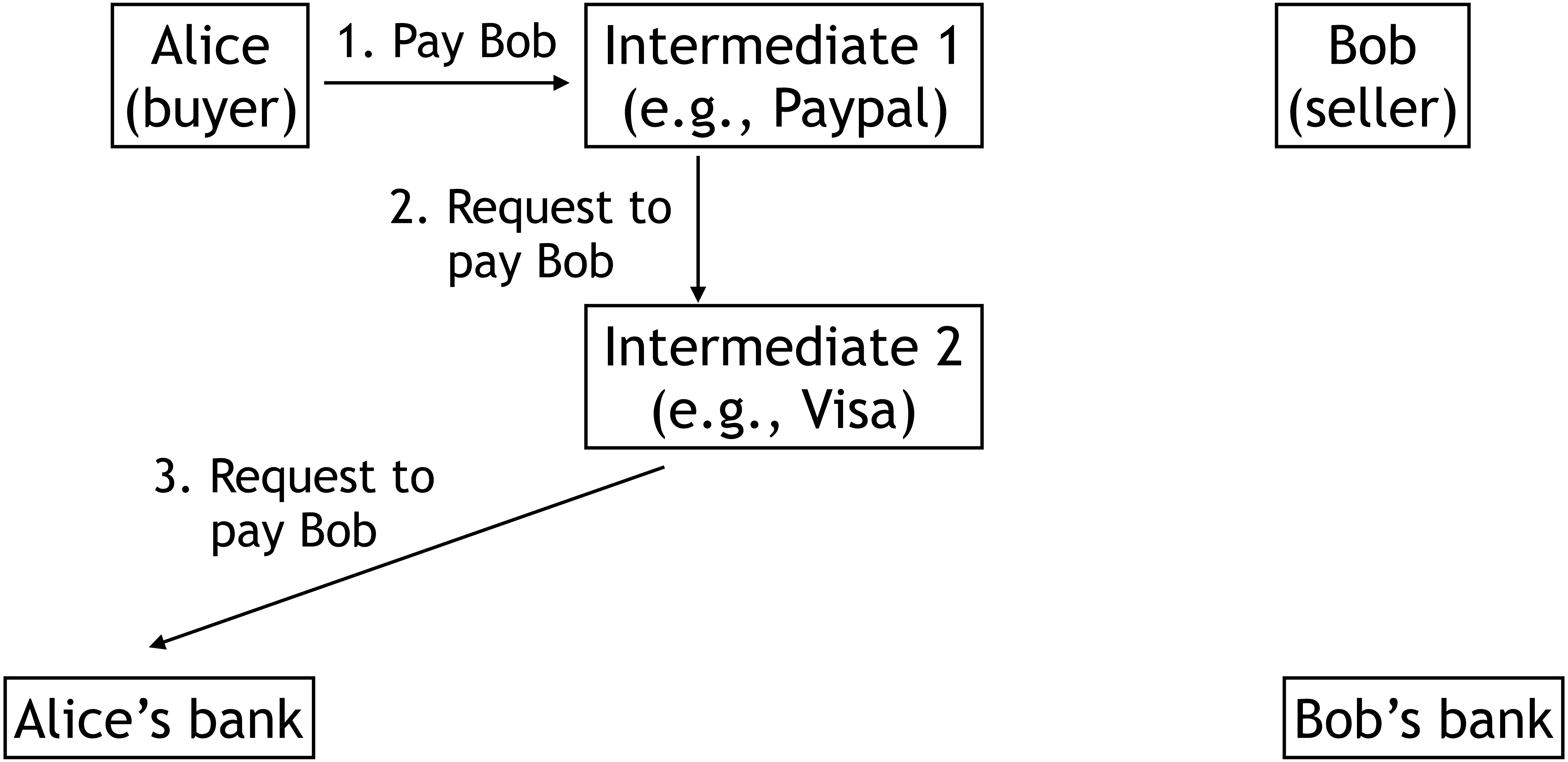
Intermediate 2 (e.g., Visa)

Bob (seller)

Alice's bank

Bob's bank

Before the bitcoin: trust



Alice (buyer)

1. Pay Bob

Intermediate 1 (e.g., Paypal)

Bob (seller)

2. Request to pay Bob

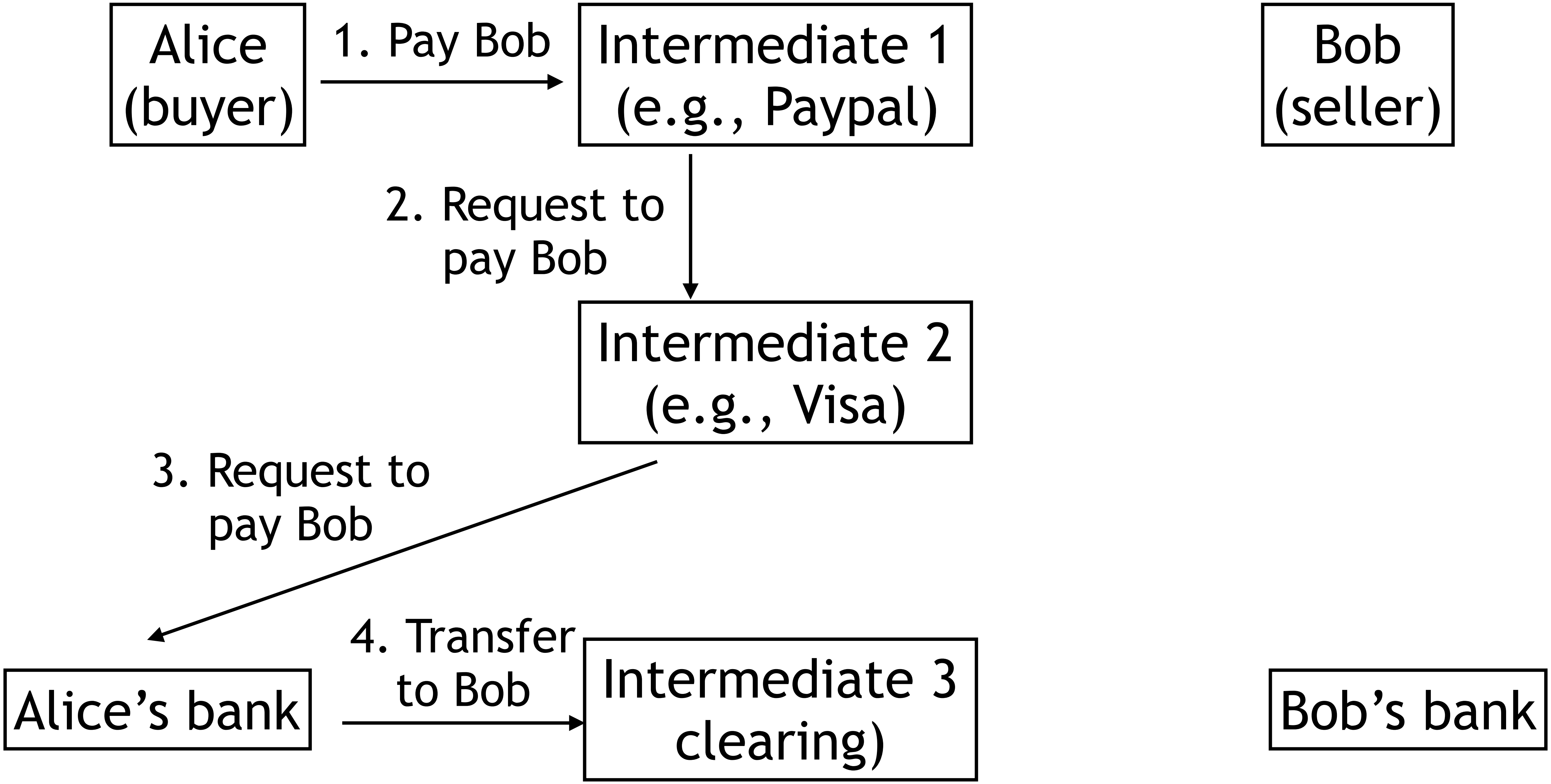
Intermediate 2 (e.g., Visa)

3. Request to pay Bob

Alice's bank

Bob's bank

Before the bitcoin: trust



Alice (buyer)

1. Pay Bob

Intermediate 1 (e.g., Paypal)

Bob (seller)

2. Request to pay Bob

Intermediate 2 (e.g., Visa)

3. Request to pay Bob

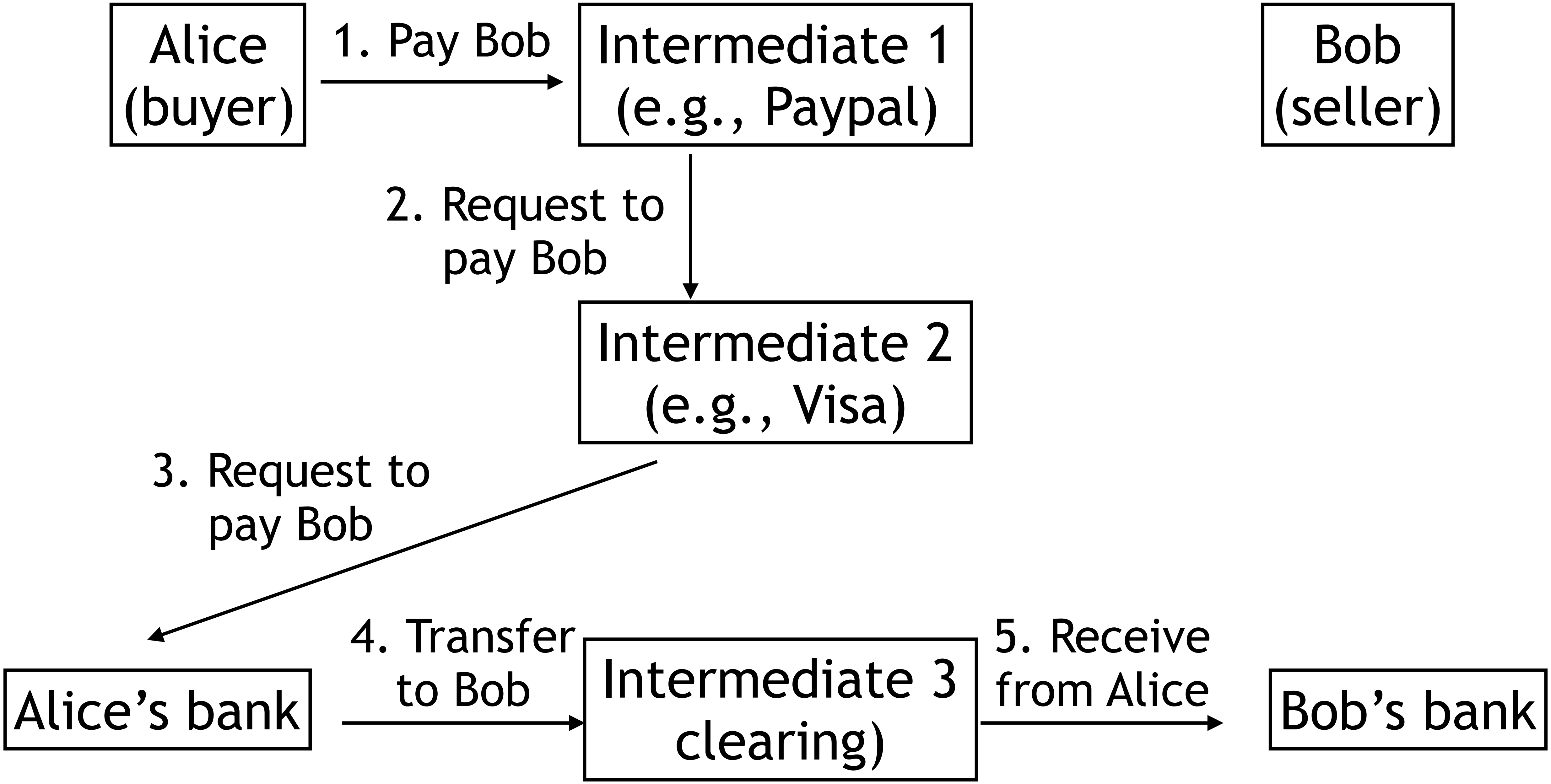
Alice's bank

4. Transfer to Bob

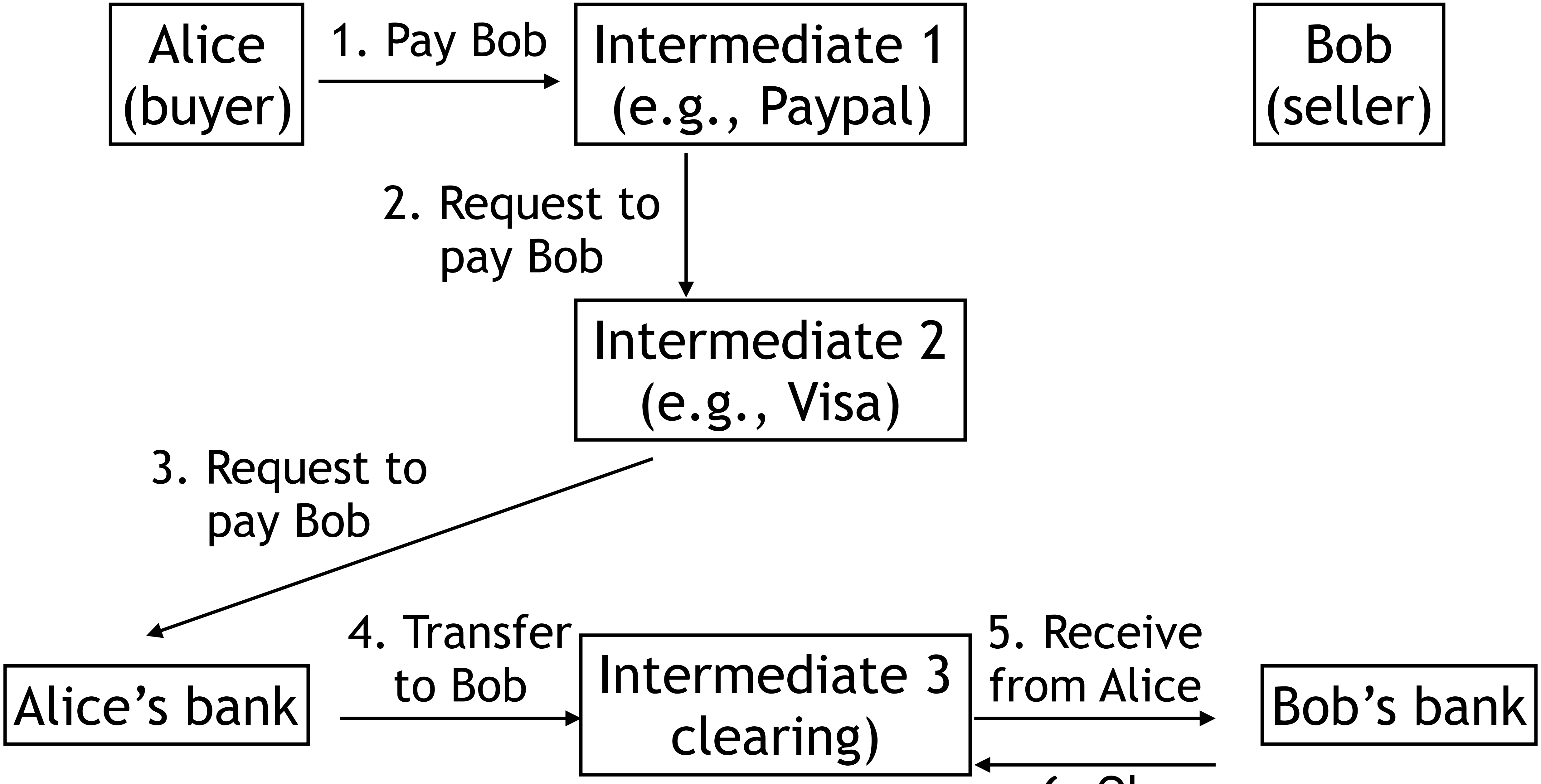
Intermediate 3 clearing)

Bob's bank

Before the bitcoin: trust



Before the bitcoin: trust



Alice (buyer)

1. Pay Bob

Intermediate 1 (e.g., Paypal)

Bob (seller)

2. Request to pay Bob

Intermediate 2 (e.g., Visa)

3. Request to pay Bob

Alice's bank

4. Transfer to Bob

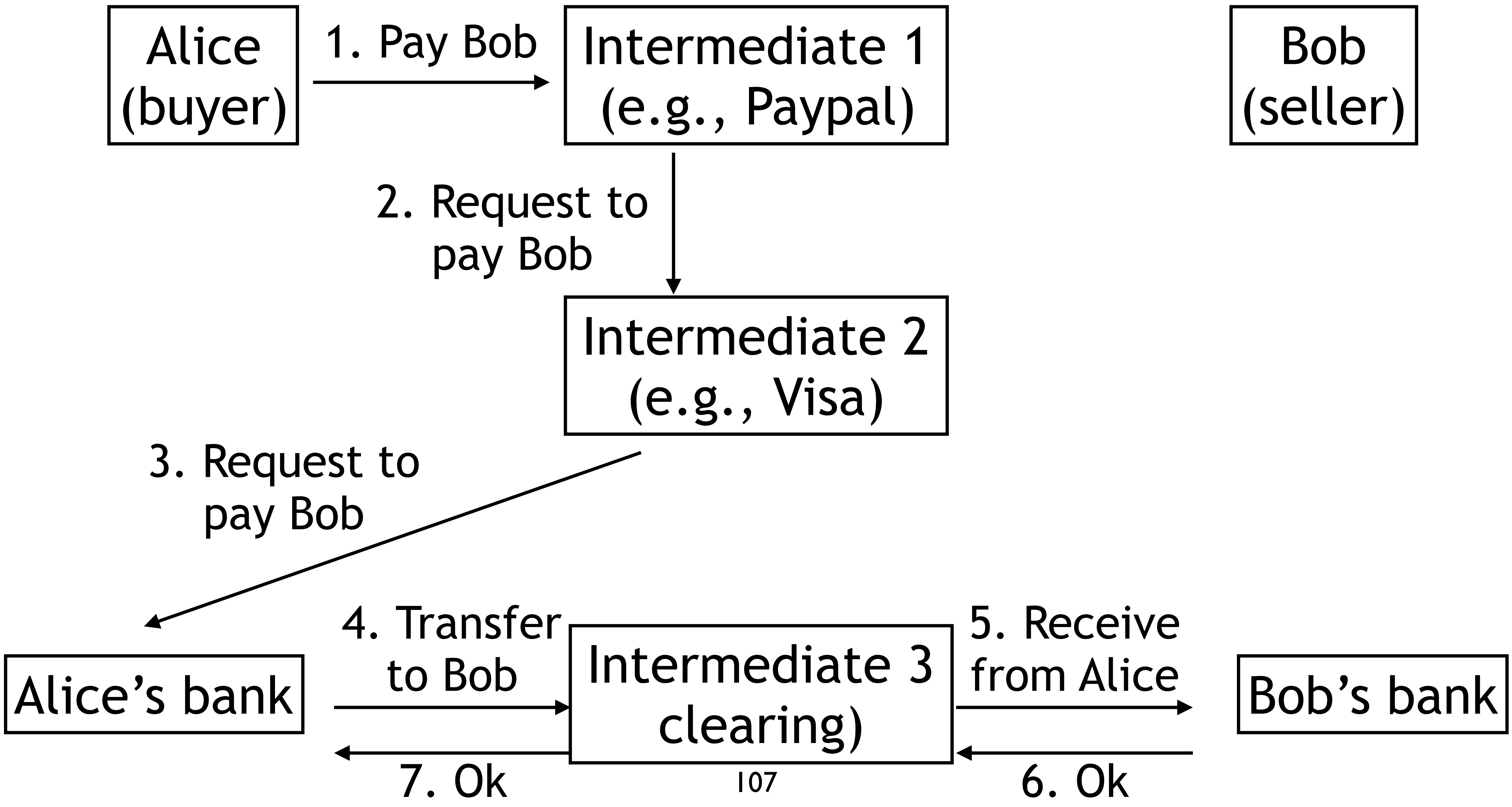
Intermediate 3 clearing)

5. Receive from Alice

Bob's bank

6. Ok

Before the bitcoin: trust



Alice (buyer)

1. Pay Bob

Intermediate 1 (e.g., Paypal)

Bob (seller)

2. Request to pay Bob

Intermediate 2 (e.g., Visa)

3. Request to pay Bob

Alice's bank

4. Transfer to Bob

Intermediate 3 clearing)

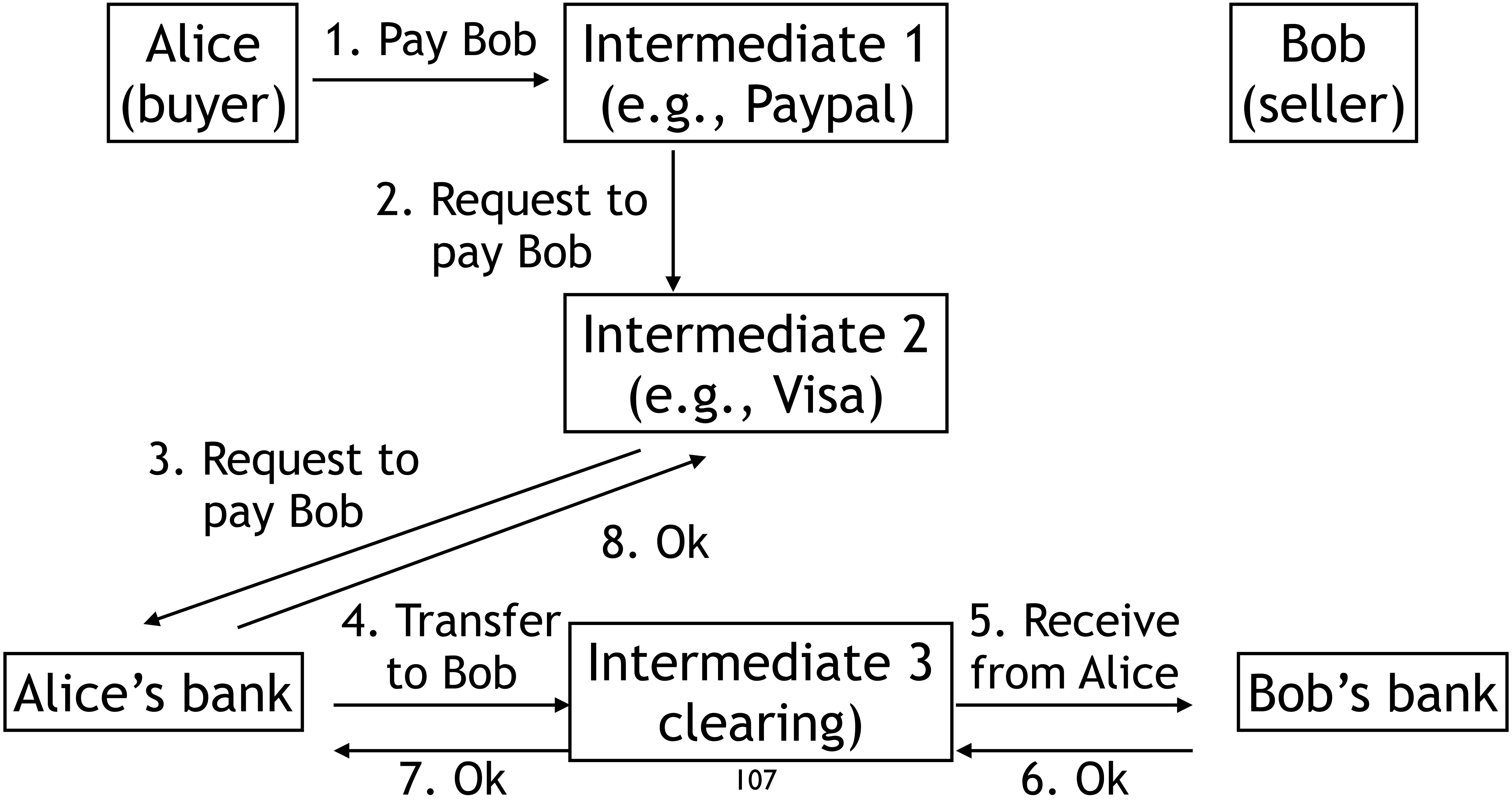
5. Receive from Alice

Bob's bank

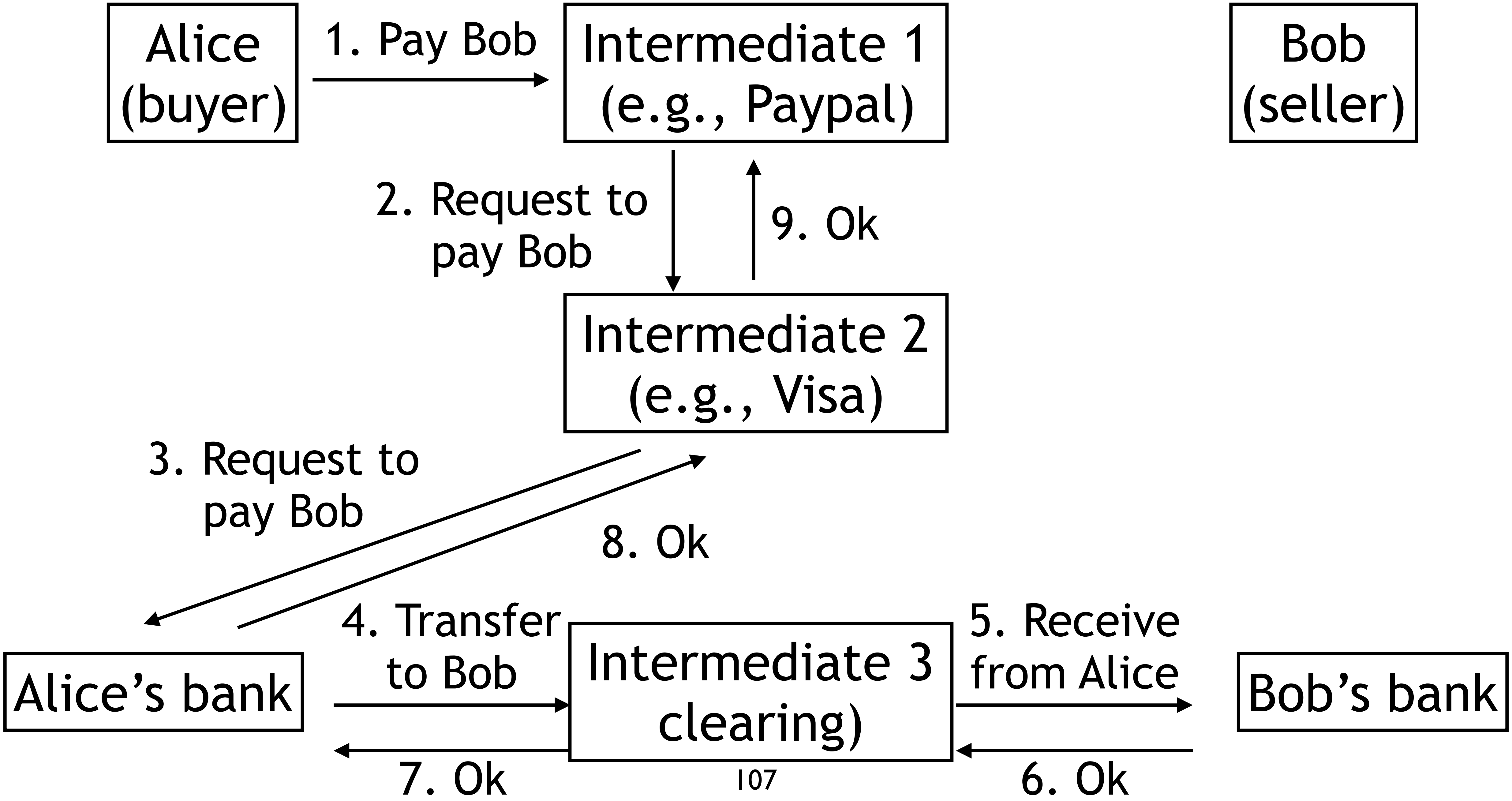
6. Ok

7. Ok

Before the bitcoin: trust



Before the bitcoin: trust



Alice (buyer)

1. Pay Bob

Intermediate 1 (e.g., Paypal)

Bob (seller)

2. Request to pay Bob

9. Ok

Intermediate 2 (e.g., Visa)

3. Request to pay Bob

8. Ok

Alice's bank

4. Transfer to Bob

Intermediate 3 clearing)

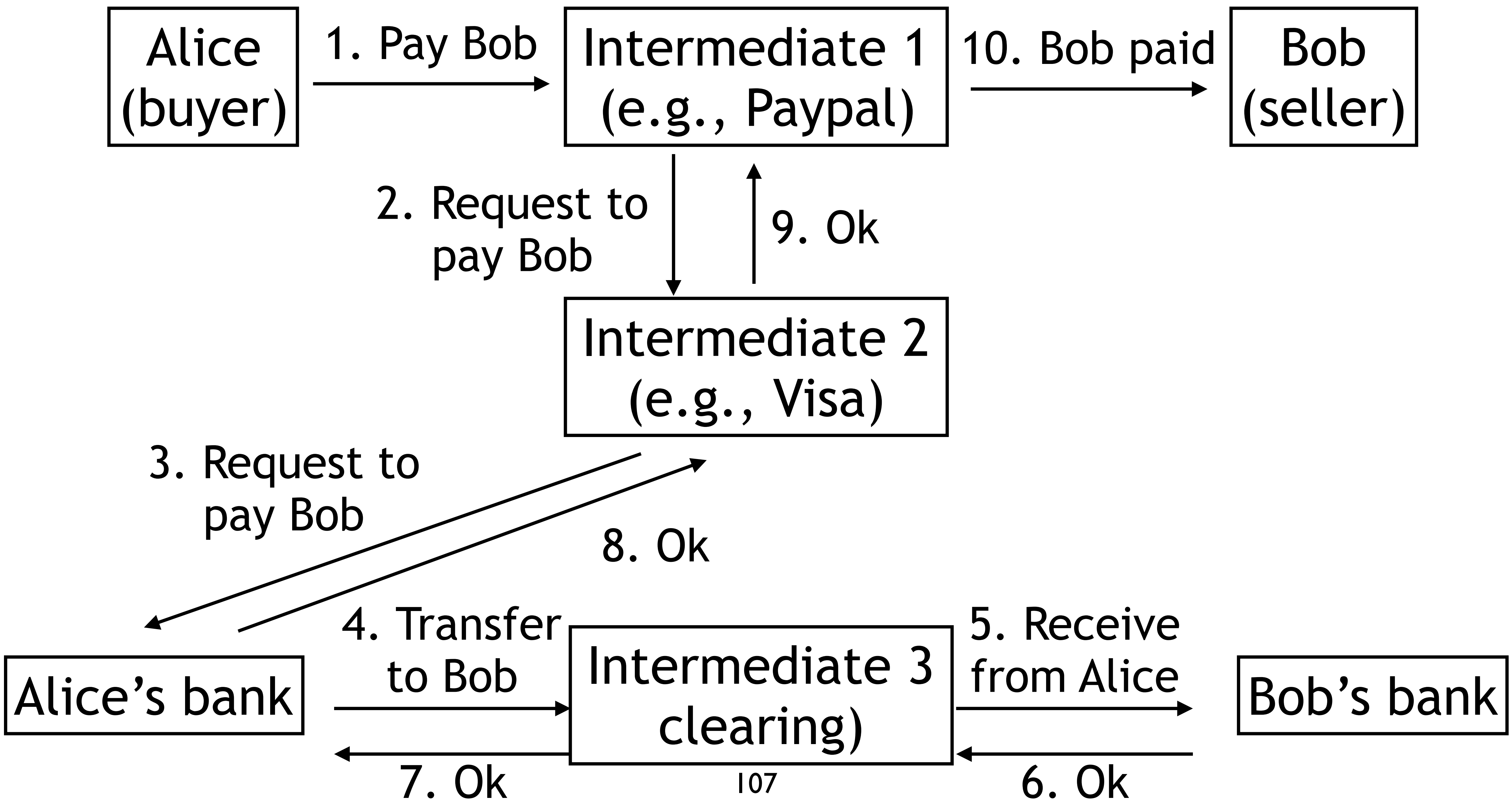
5. Receive from Alice

Bob's bank

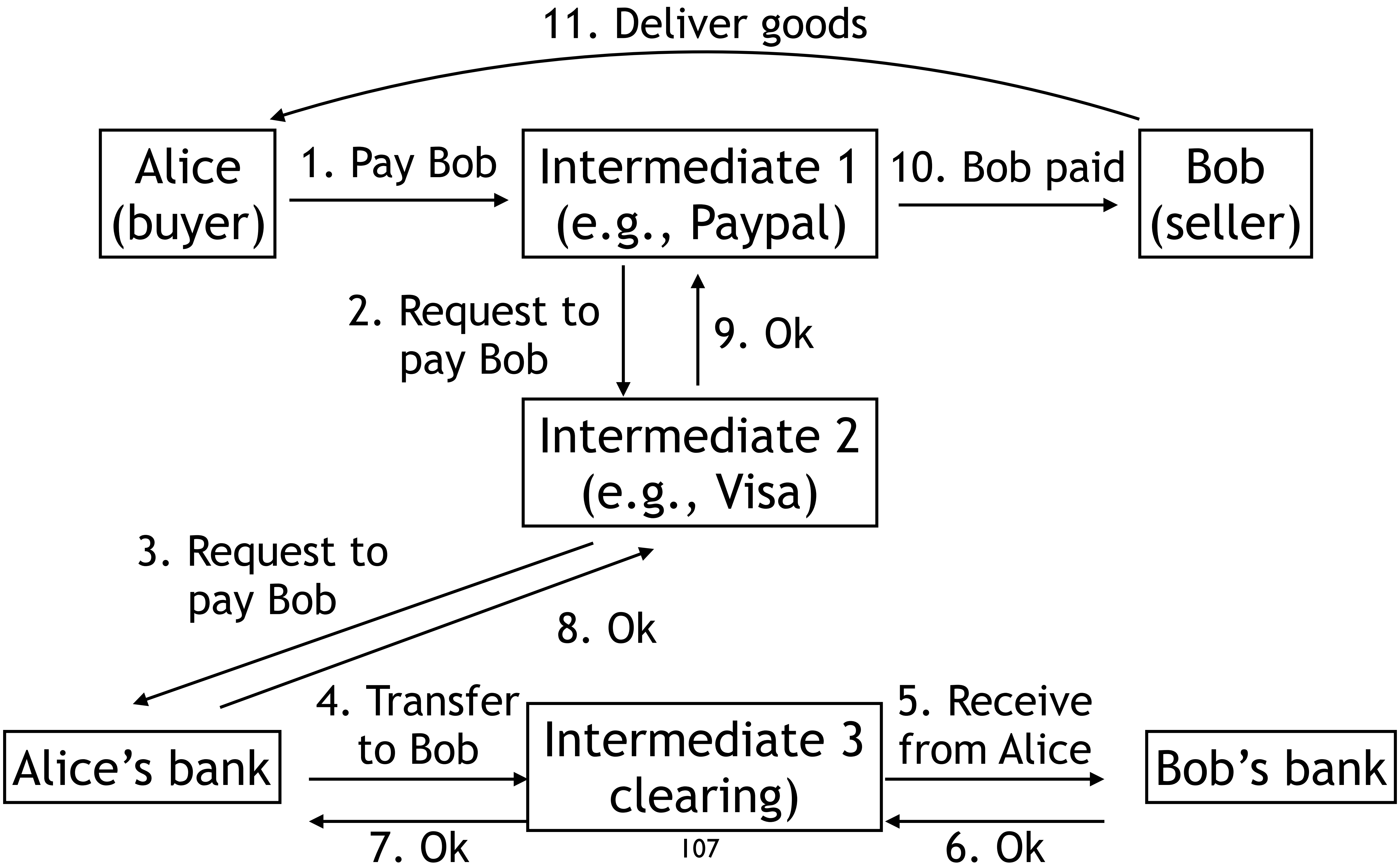
7. Ok

6. Ok

Before the bitcoin: trust



Before the bitcoin: trust



11. Deliver goods

Alice (buyer)

1. Pay Bob

Intermediate 1 (e.g., Paypal)

10. Bob paid

Bob (seller)

2. Request to pay Bob

9. Ok

Intermediate 2 (e.g., Visa)

3. Request to pay Bob

8. Ok

Alice's bank

4. Transfer to Bob

Intermediate 3 clearing)

5. Receive from Alice

Bob's bank

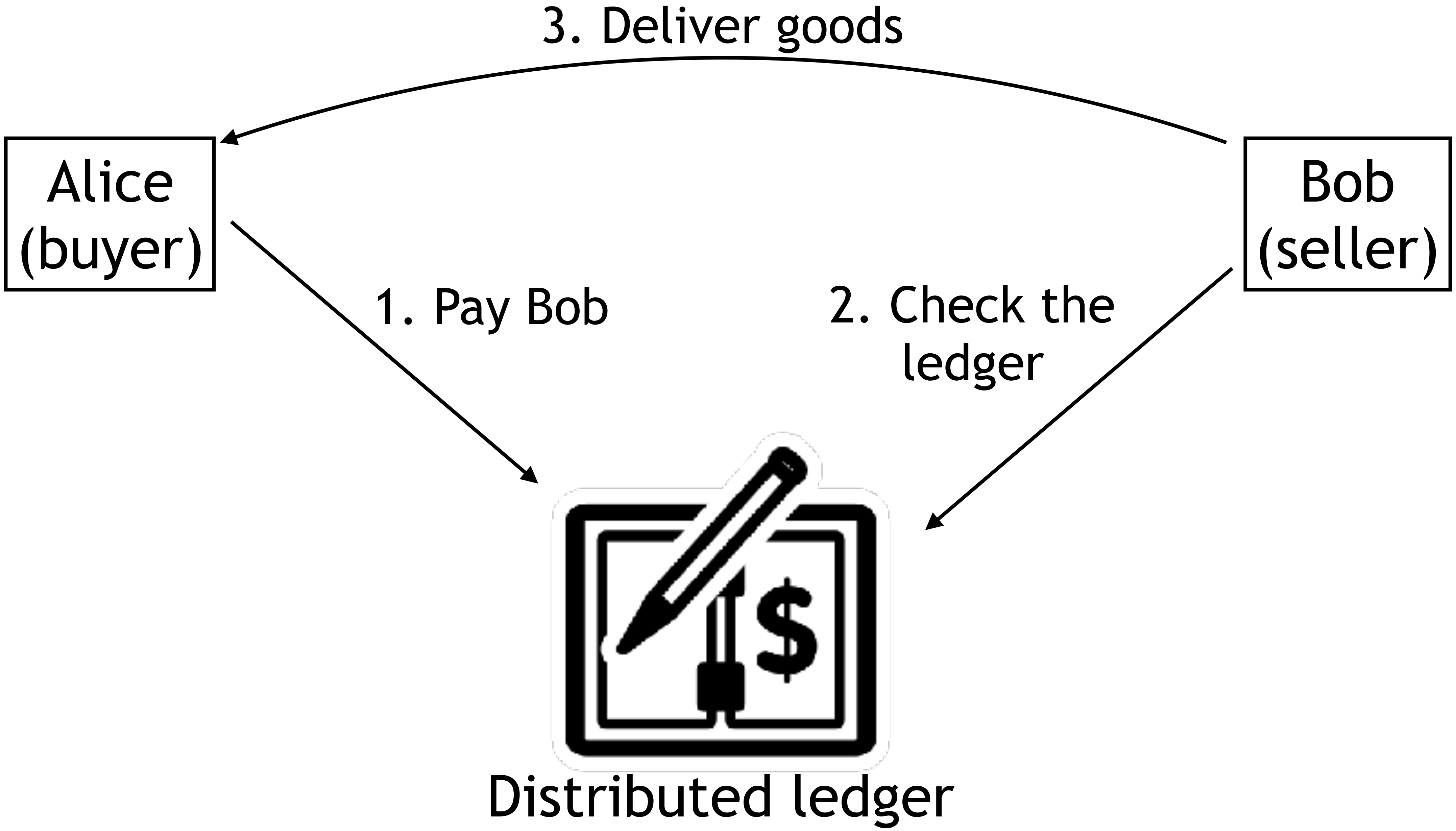
7. Ok

6. Ok

The role of intermediates: establish trust

- Alice and Bob don’t trust each other
  - need to find a common trusted party
- Banks don’t trust each other
  - clearing houses settle transactions
- Each intermediate gets their share and concentrates power

With the bitcoin: proof



3. Deliver goods

Alice (buyer)

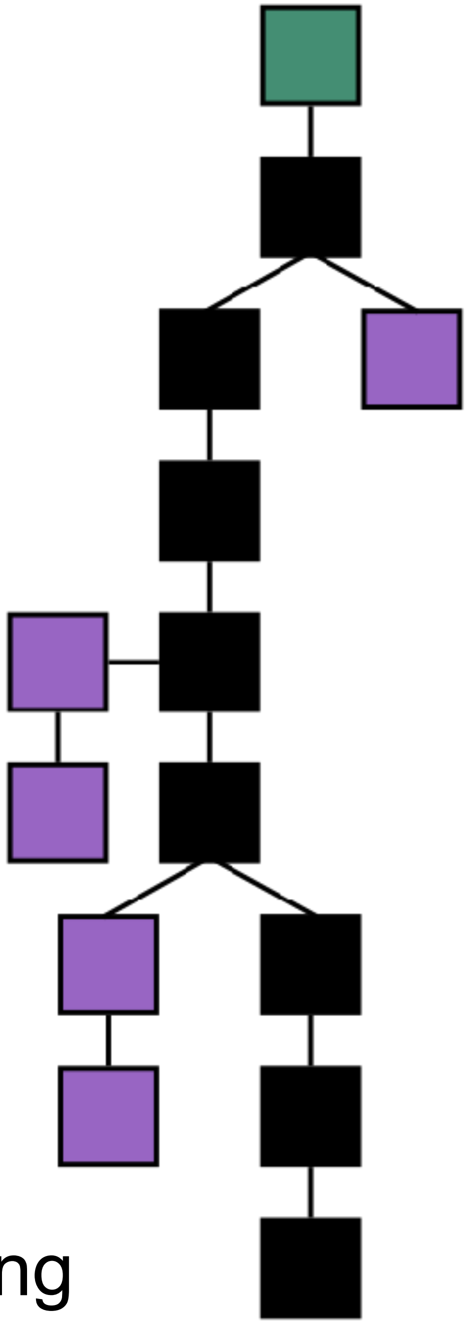
Bob (seller)

1. Pay Bob

2. Check the ledger

Distributed ledger

Overview



- The blockchain is a list of blocks
  - a block is associated to its cryptographic hash that encompasses
    - the block data
    - the block timestamp
    - the block nonce
    - the hash of the predecessor in the list
      - blocks are “cryptographically linked” preventing them to be tempered
      - the blocks chronology is guaranteed

Overview (contd.)

- The list of blocks is distributed in the network
    - using a peer-to-peer network (all nodes seem to be connected)
        - supporting broadcast
- Transactions are broadcasted in the network

Overview (contd.)

- Miners create new block based on the collect transactions
  - a new block is added only if the majority of miners agree
  - every miner collects broadcasted transactions
    - and groups them together to form the data of the block
    - when enough transactions are in the block, the miner computes a valid hash
    - and broadcasts it to the network
  - the first broadcasted new valid block is added as the new head of the blockchain, the fastest miner is the winner
    - the winner is rewarded by gaining some fraction of bitcoin

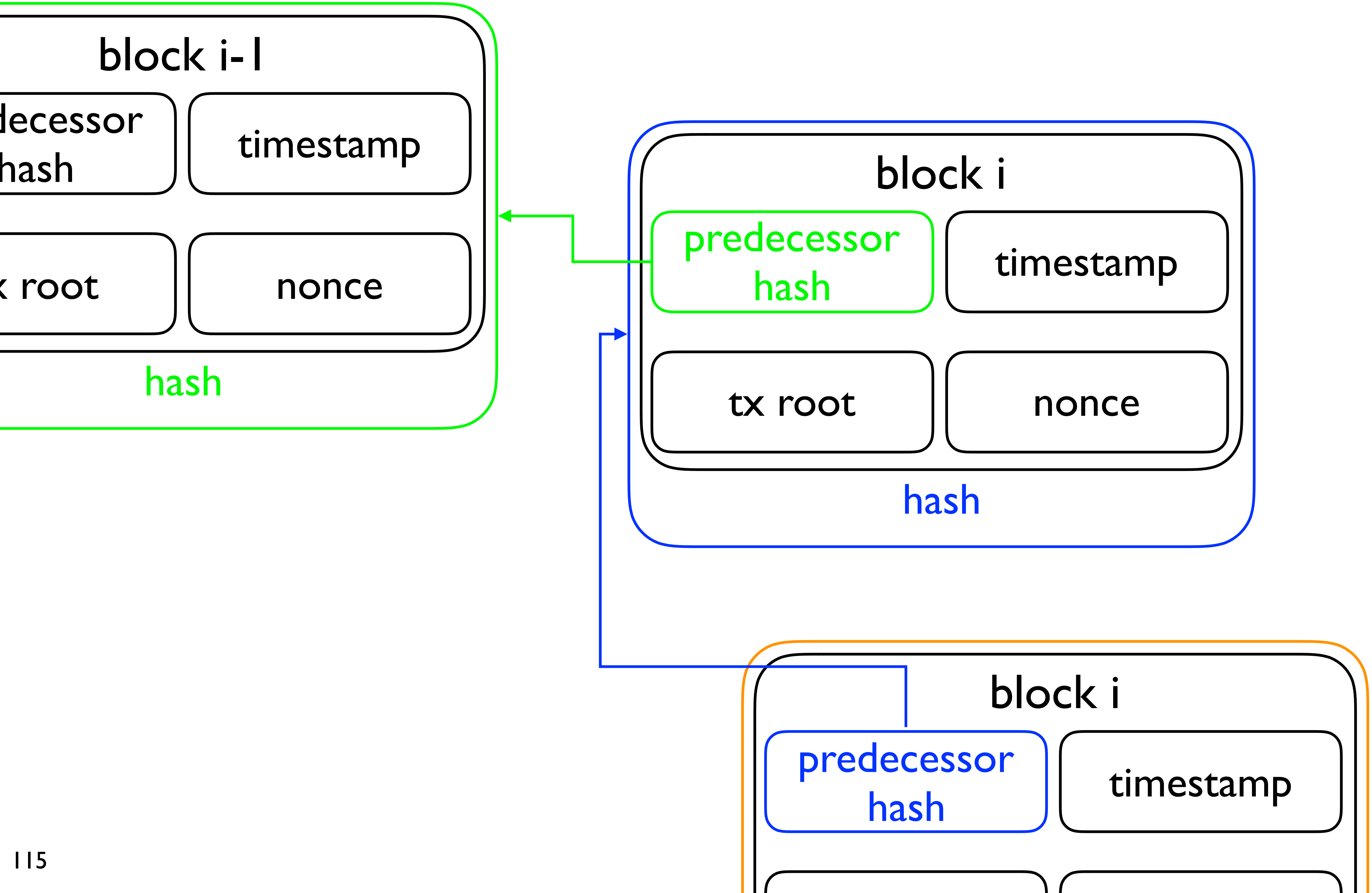
Making a transactions

- The origin of the transaction
  - adds its bitcoin address
  - adds the bitcoin address of the destination
  - signs the transaction using its private key
  - advertises it in the network
- Anyone can verify the origin of the transaction using the public key
  - and its presence in the blockchain

How to identity clients in bitcoin?

- Clients have a wallet
  - the wallet is just a private/public key pair
- Identification with a bitcoin address
  - generated for free by any bitcoin user
    - public key = elliptic curve multiplication of the private key
    - bitcoin address = hash of the public key
      - represented with a 26-35 alphanumeric value

Inside a block



Merkel tree

- Transactions are stored in a Merkel tree
- In a Merkel tree
  - the key of a node is the hash of its two children
  - except for the leaves where it is the hash of the data itself
    - in bitcoin, the hash is the SHA-256 hash of the SHA-256 hash of the item to hash

Merkel tree (contd.)

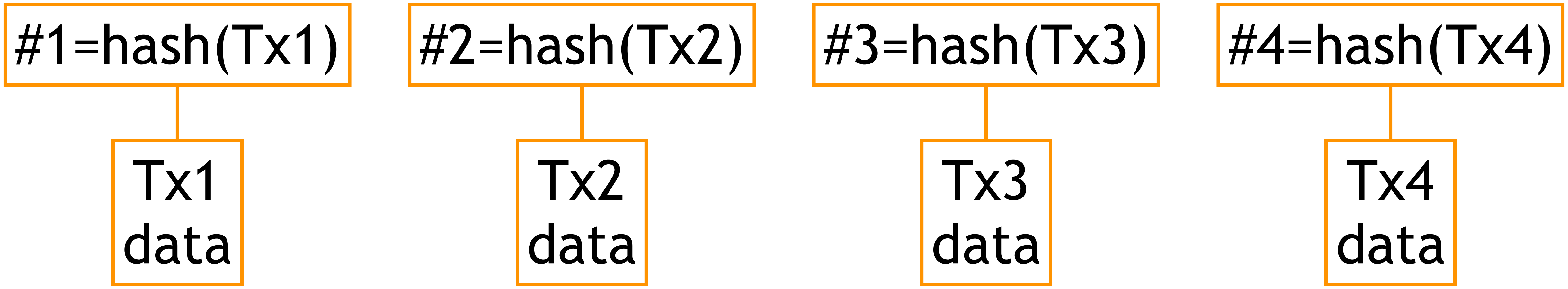
Tx1 data

Tx2 data

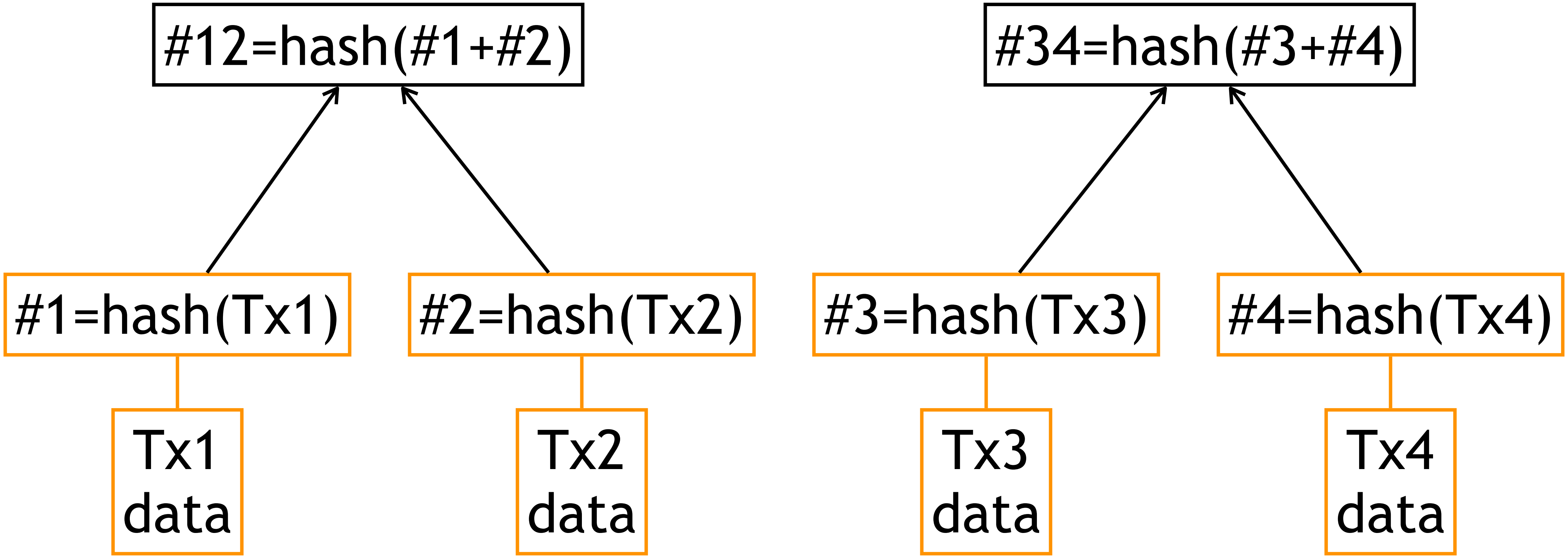
Tx3 data

Tx4 data

Merkel tree (contd.)



Merkel tree (contd.)



#12=hash(#1+#2)

#34=hash(#3+#4)

#1=hash(Tx1)

#2=hash(Tx2)

#3=hash(Tx3)

#4=hash(Tx4)

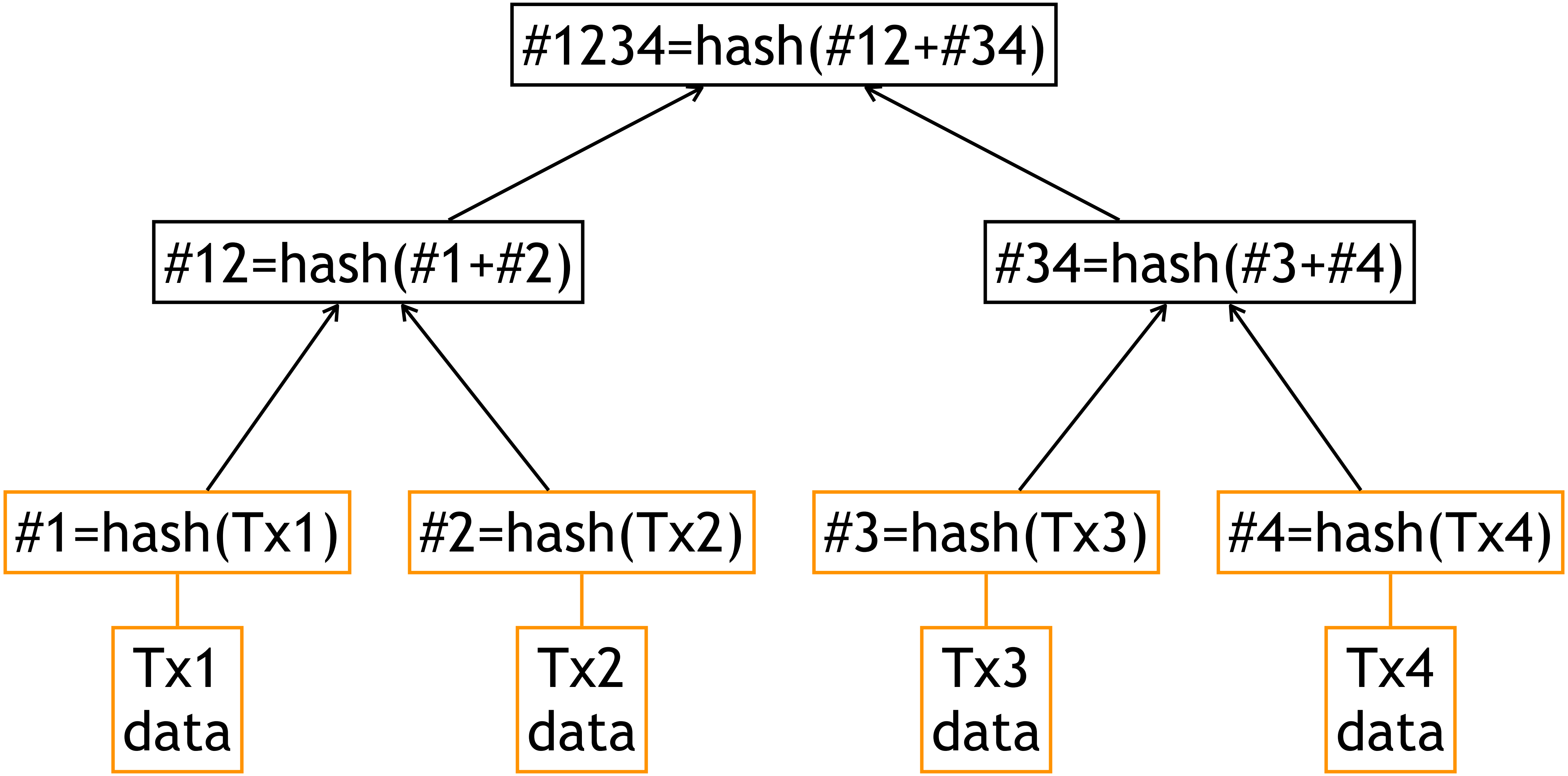
Tx1 data

Tx2 data

Tx3 data

Tx4 data

Merkel tree (contd.)

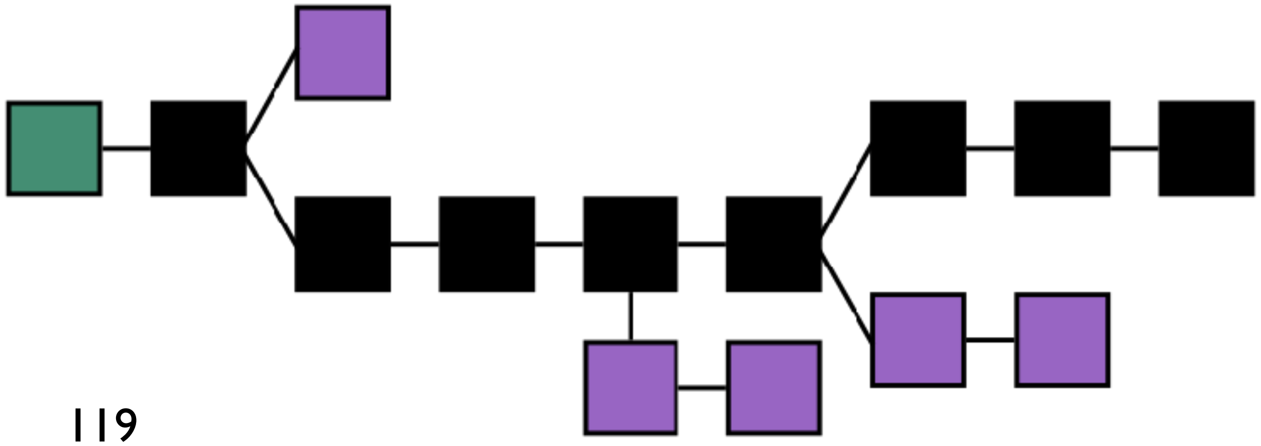


Proof of work

- To be accepted, minors must accomplish a proof of work (PoW) on the blocks they advertise
- The PoW is hard to make, easy to check
  - e.g., find a nonce such that the hash of the block is below some target value
    - the target is chosen such that the PoW takes about 10 minutes

Branch selection

- Multiple branches can be valid (e.g., two minors gave a valid block at the same time)
  - the longest (in terms of complexity) valid branch is selected
  - a block is valid if it has at least 6 successors

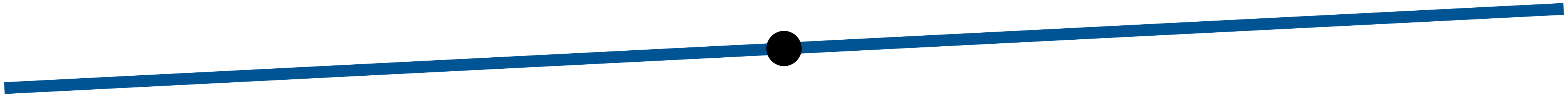


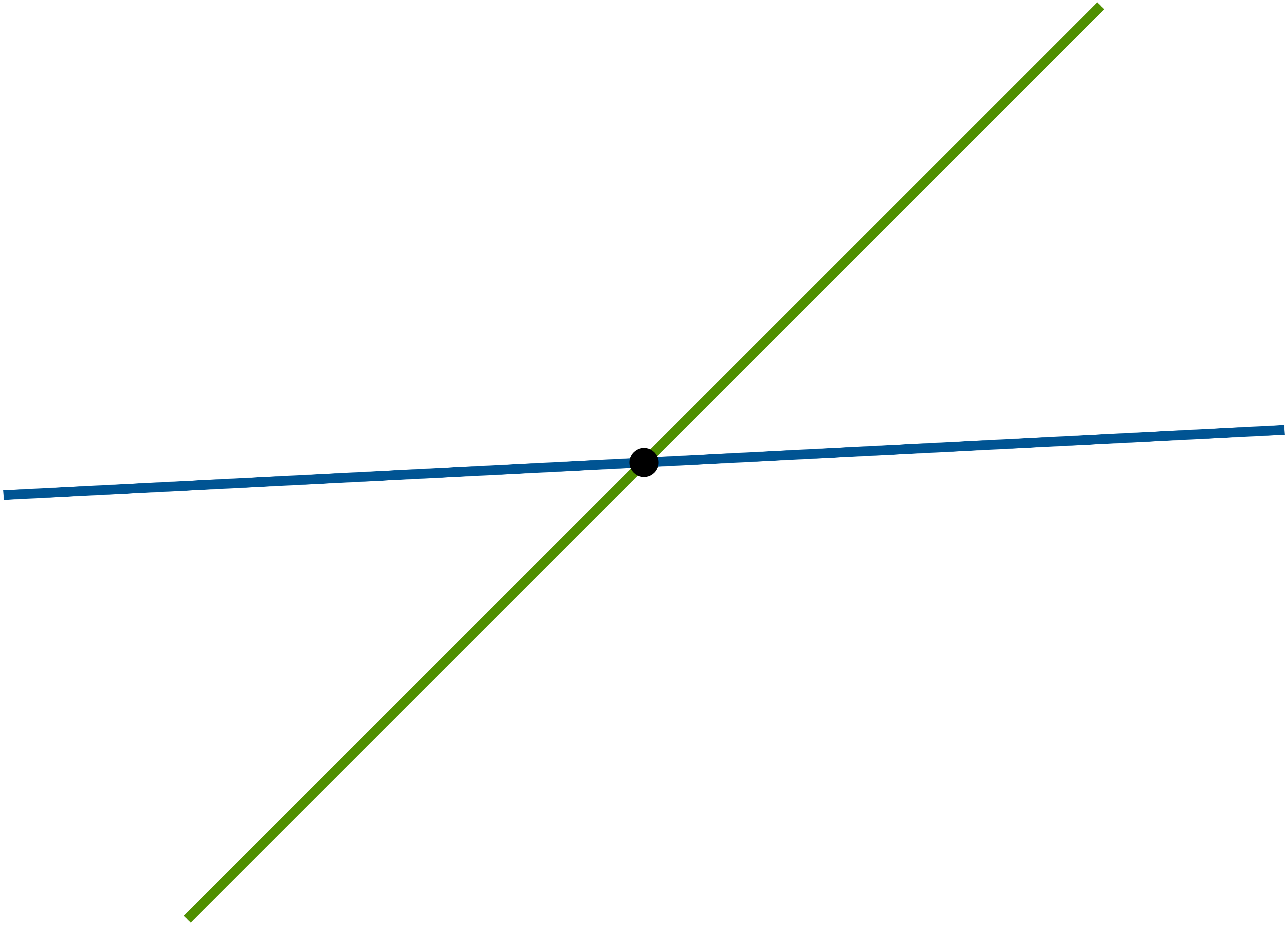
Privacy

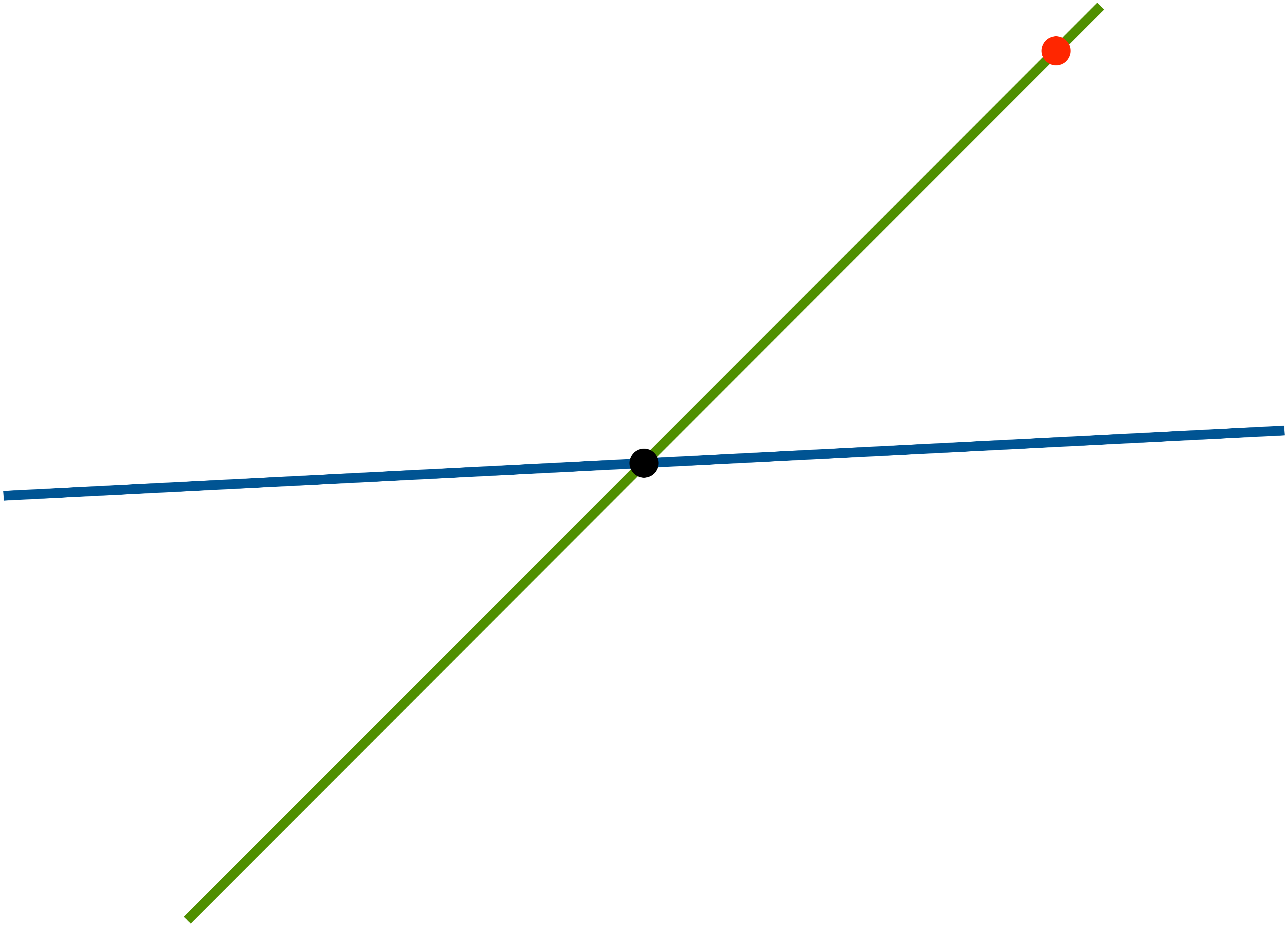
Sharing secrets

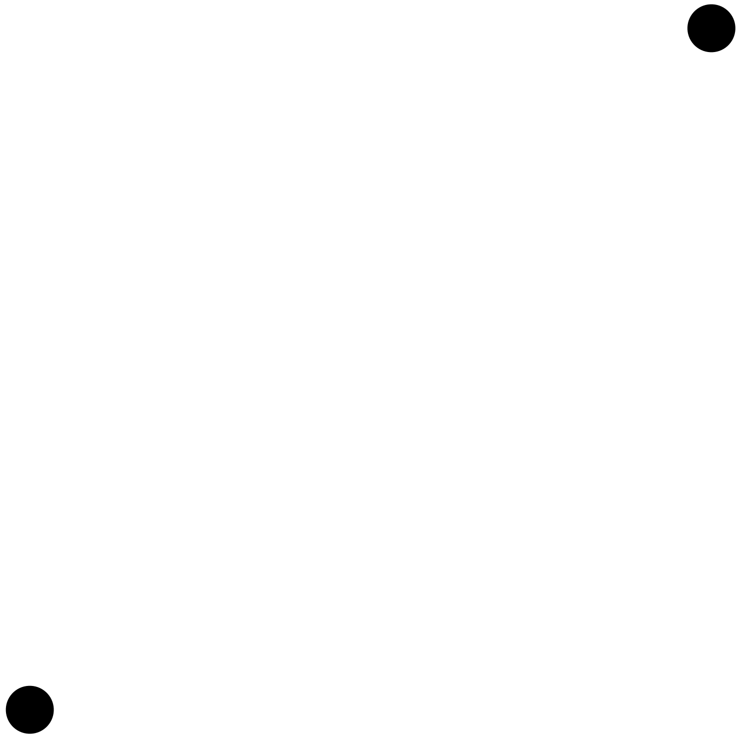
- Context
  - *n* students work on a top-secret project
  - They cannot trust each other
  - The project is in a digital safe
  - To open the digital safe, at least *k* out of the *n* students must be present

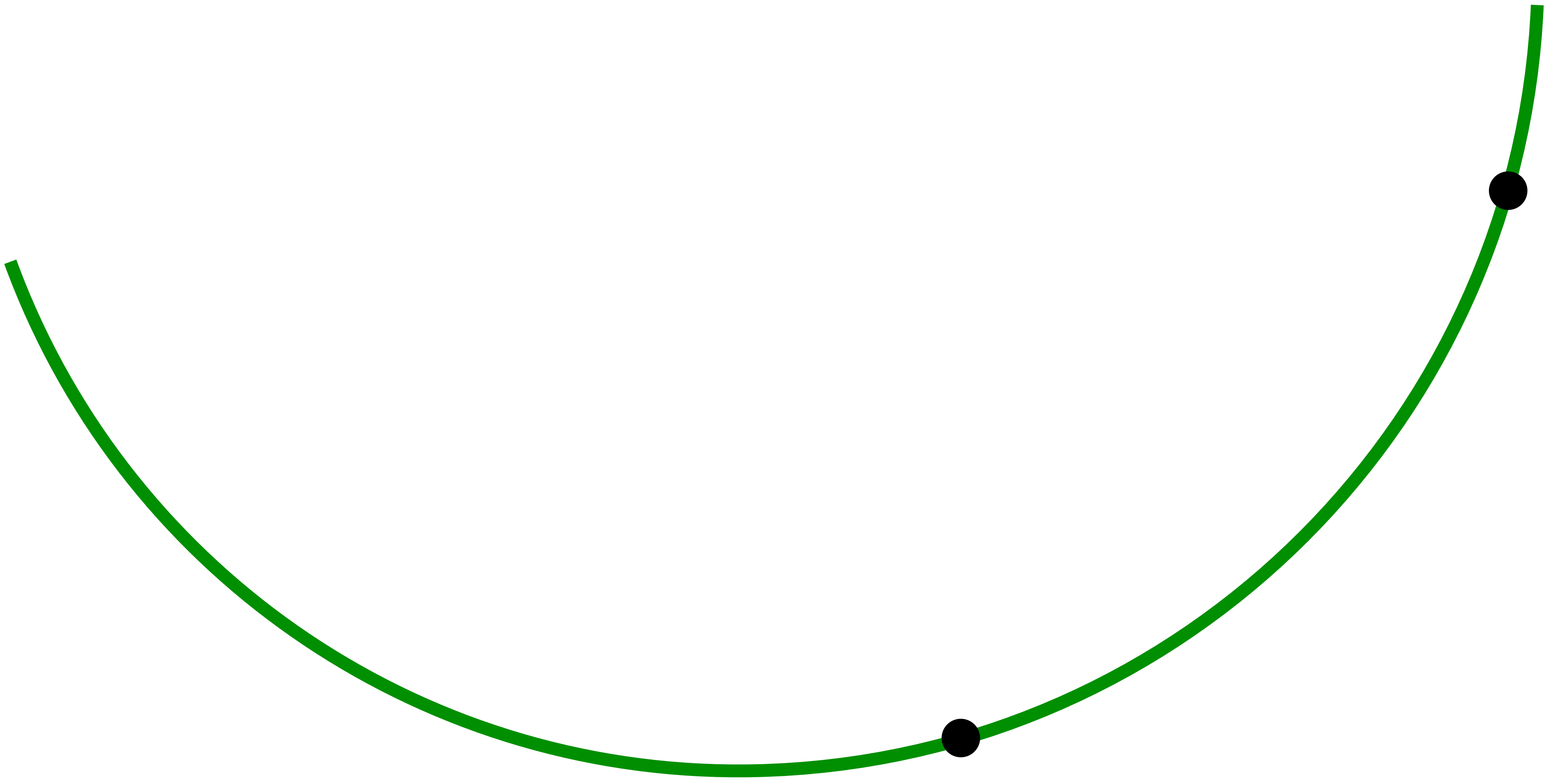
●

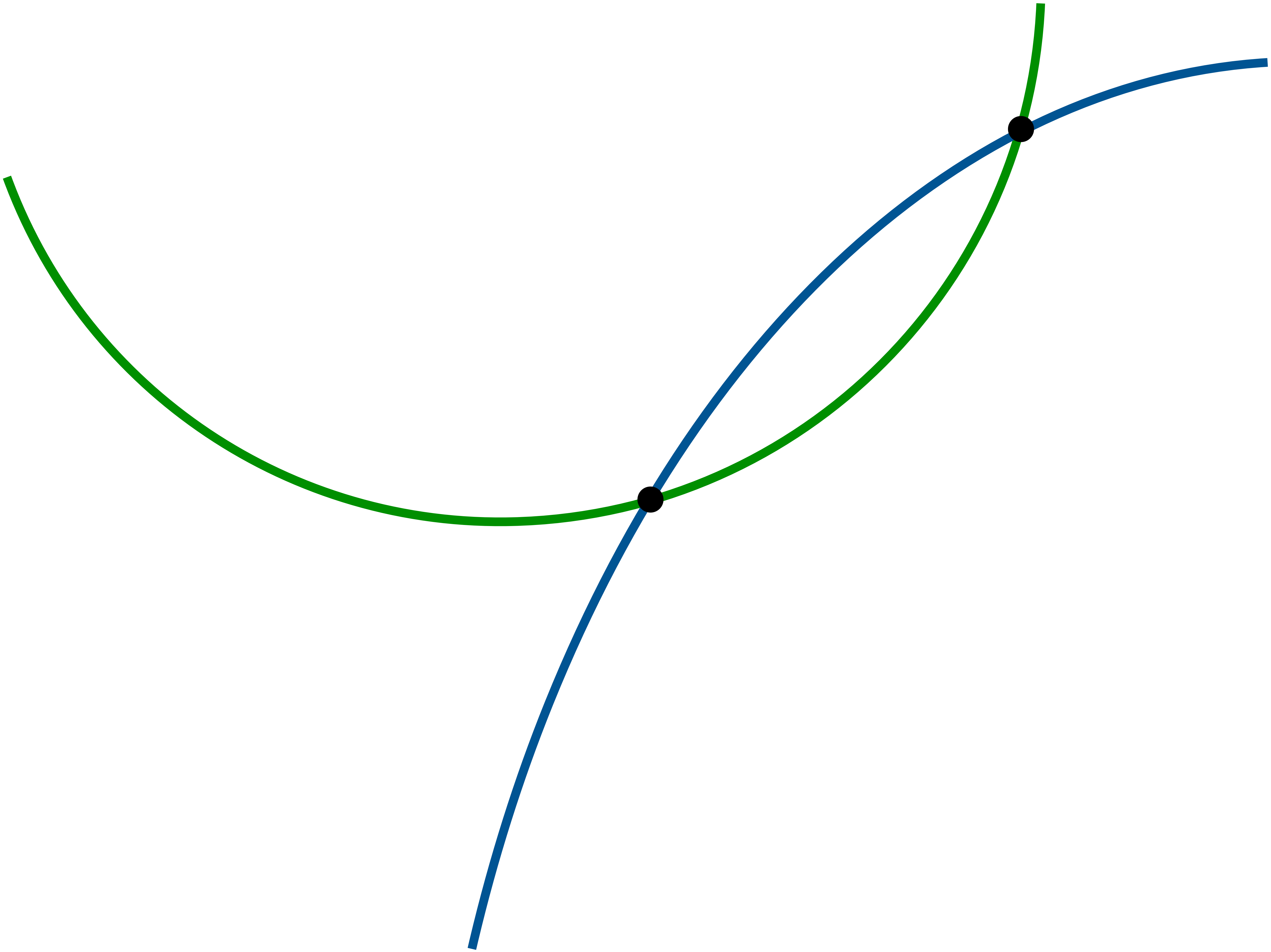


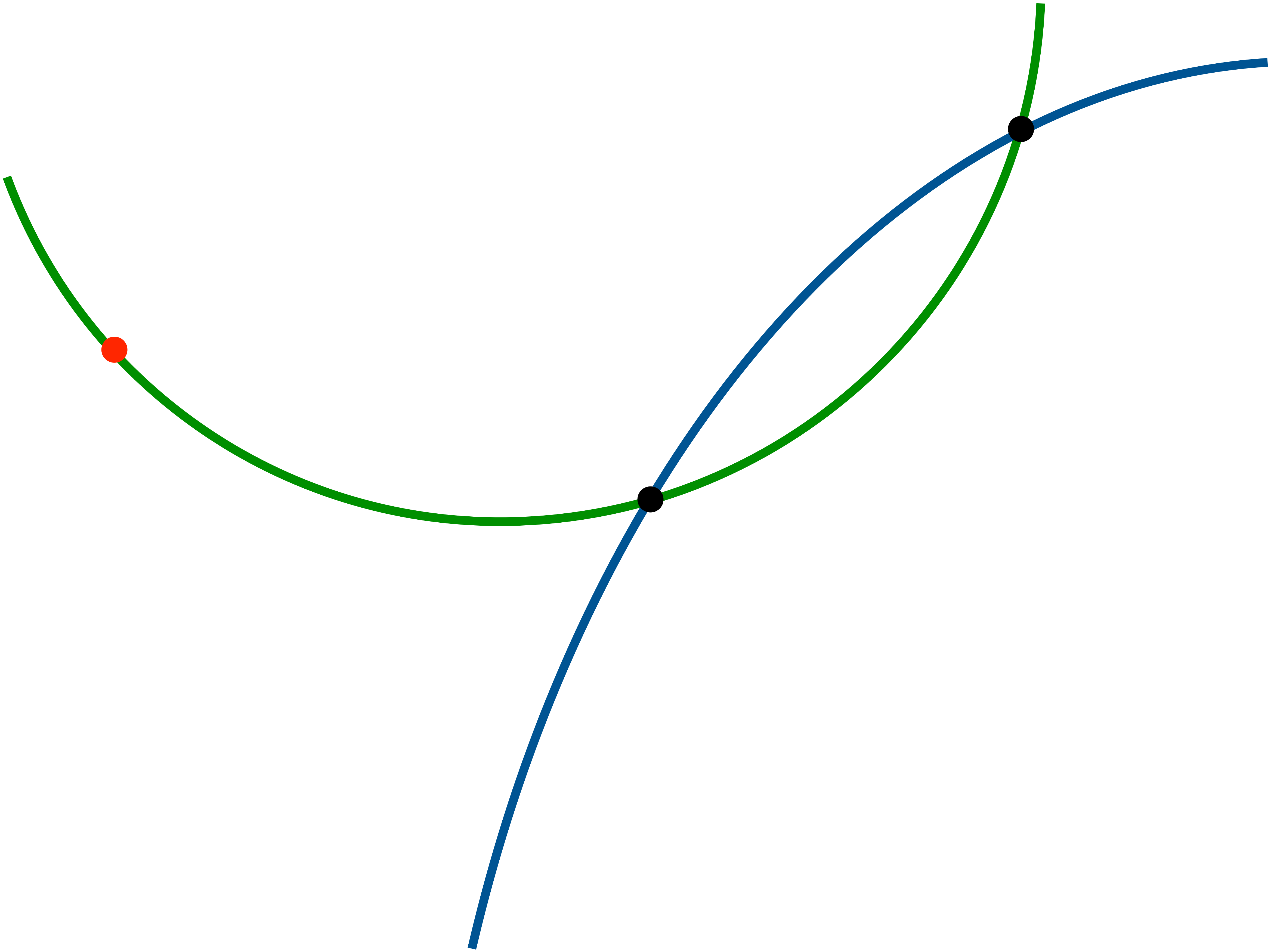


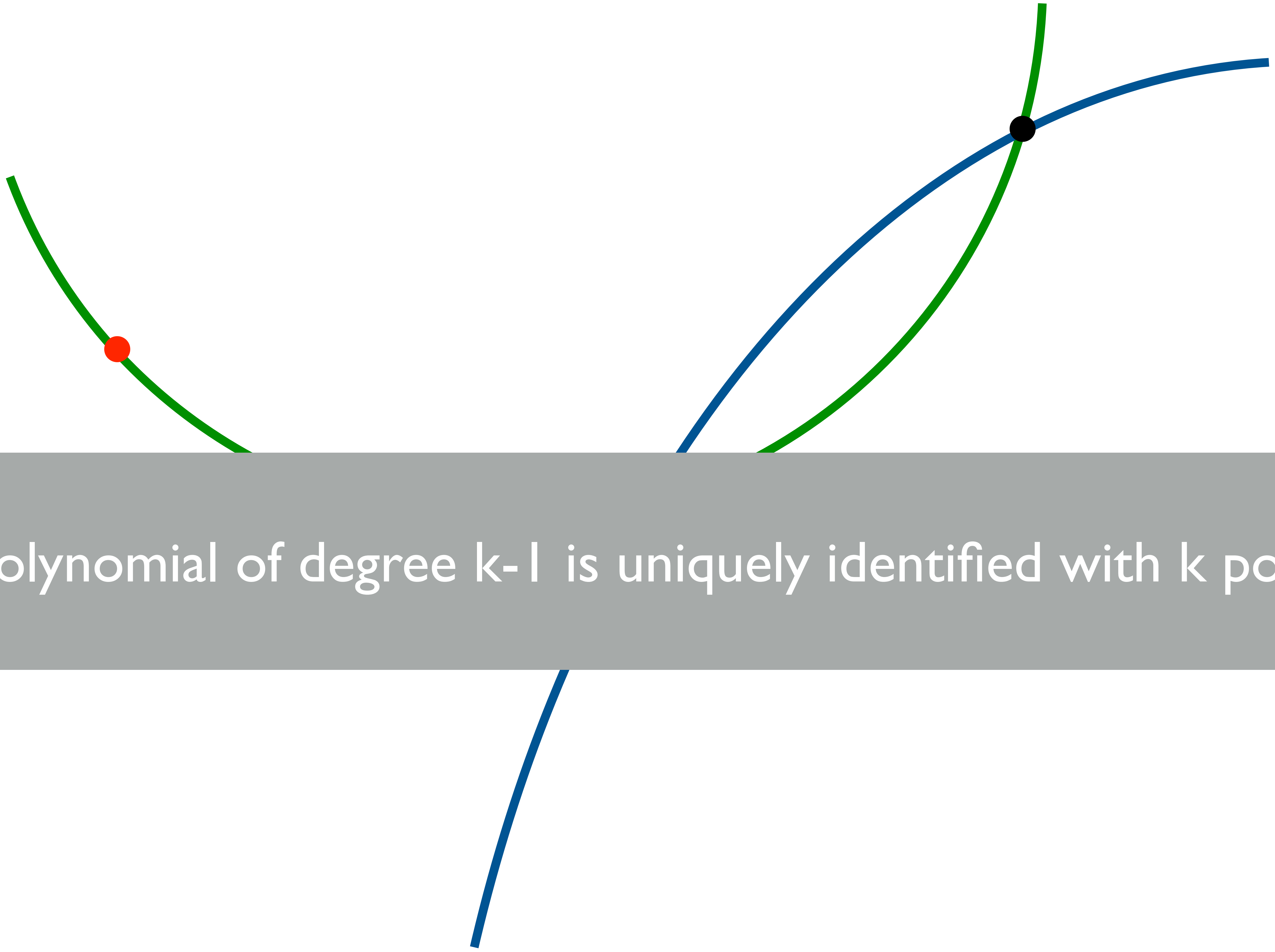










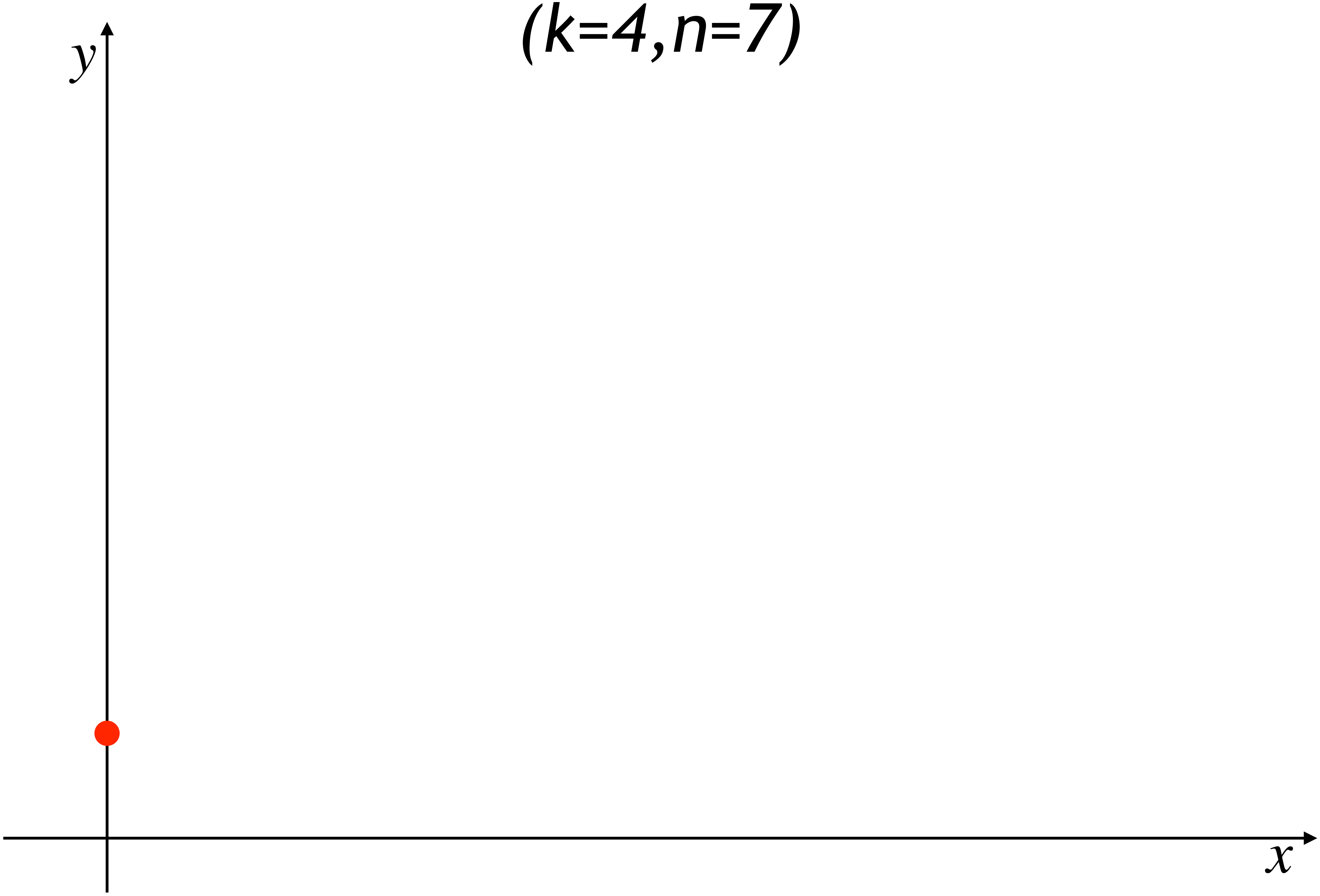


A polynomial of degree k-1 is uniquely identified with k points

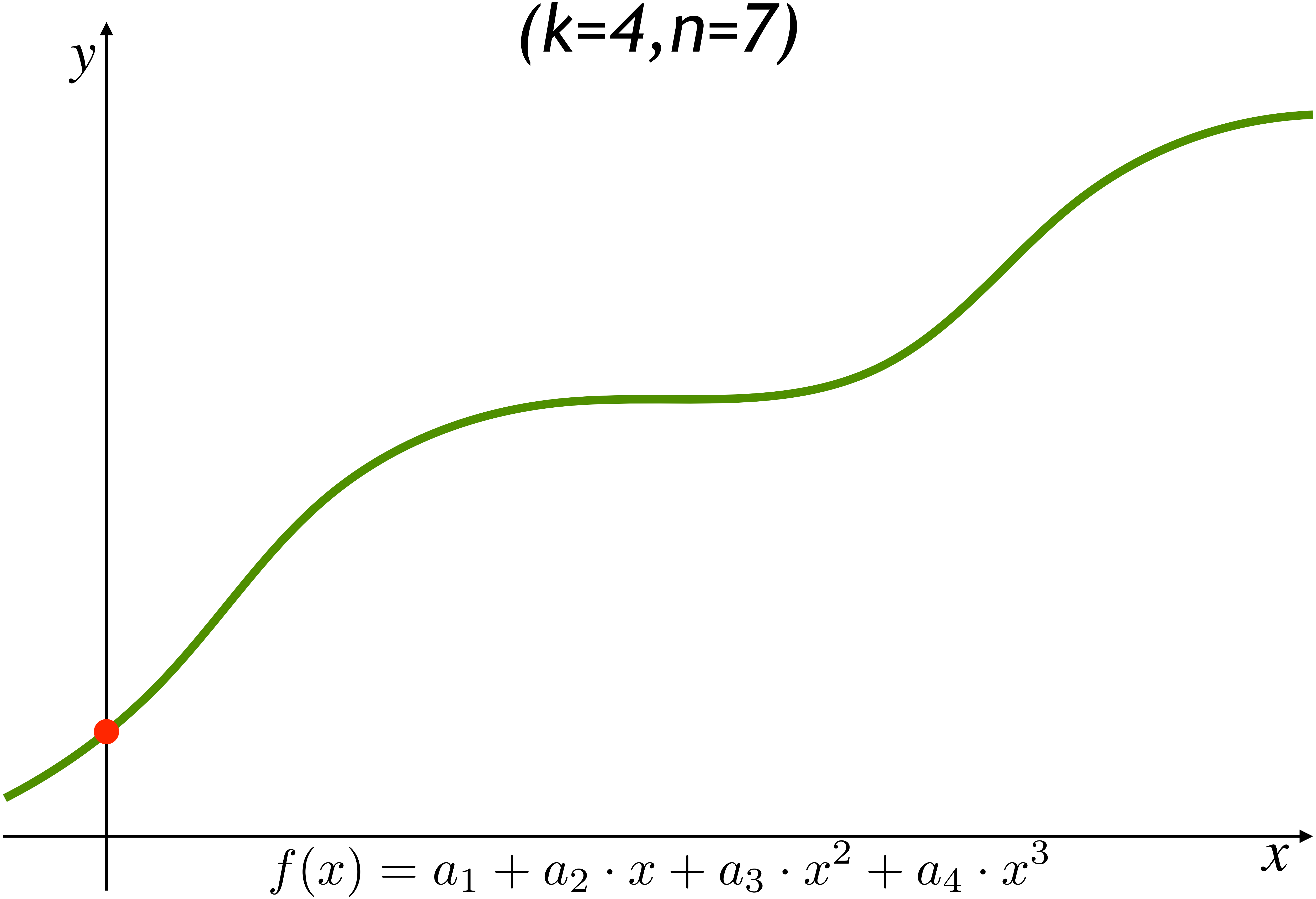
$(k=4, n=7)$

$(k=4, n=7)$

$(k{=}4, n{=}7)$

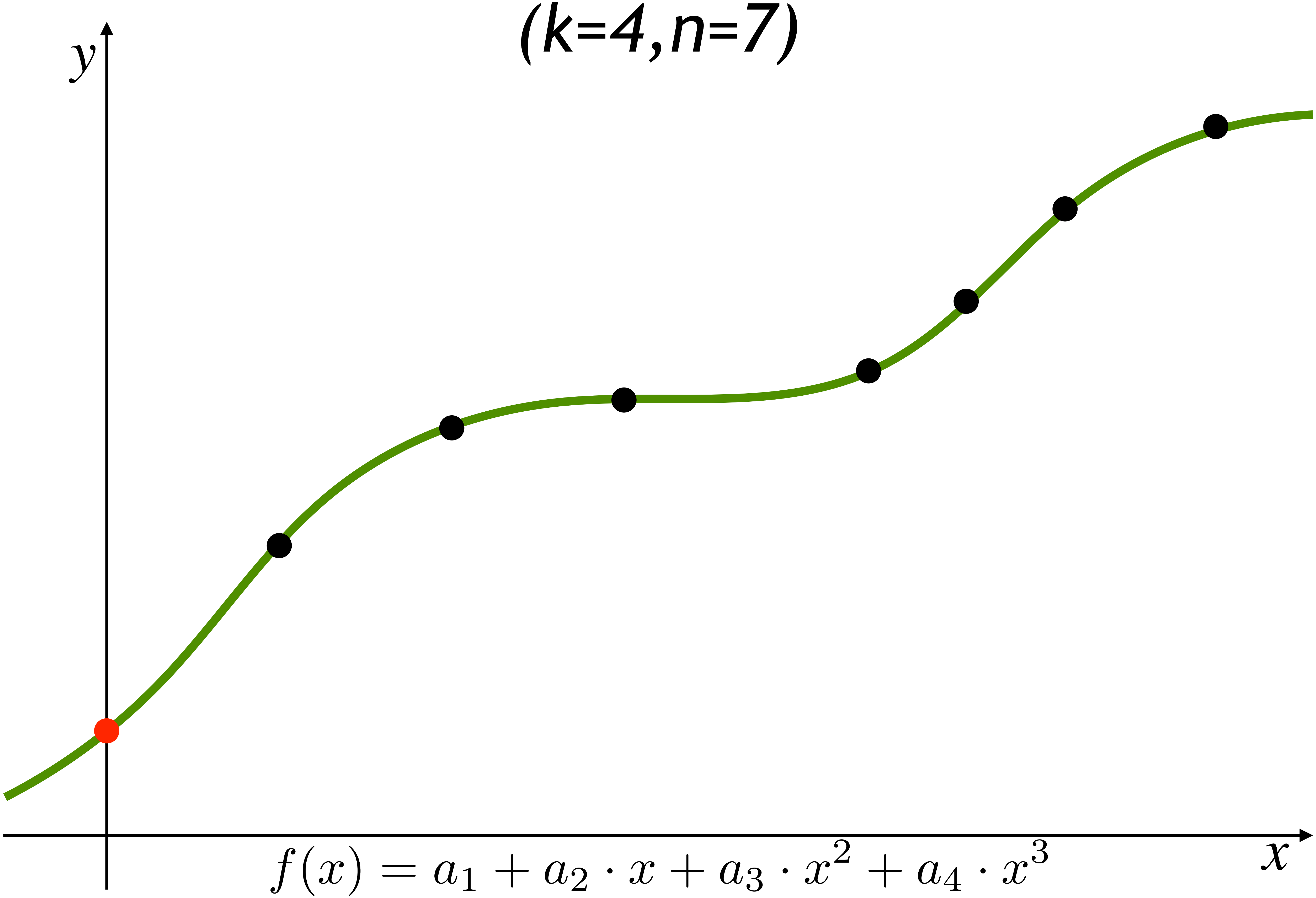


(*k*=4,*n*=7)



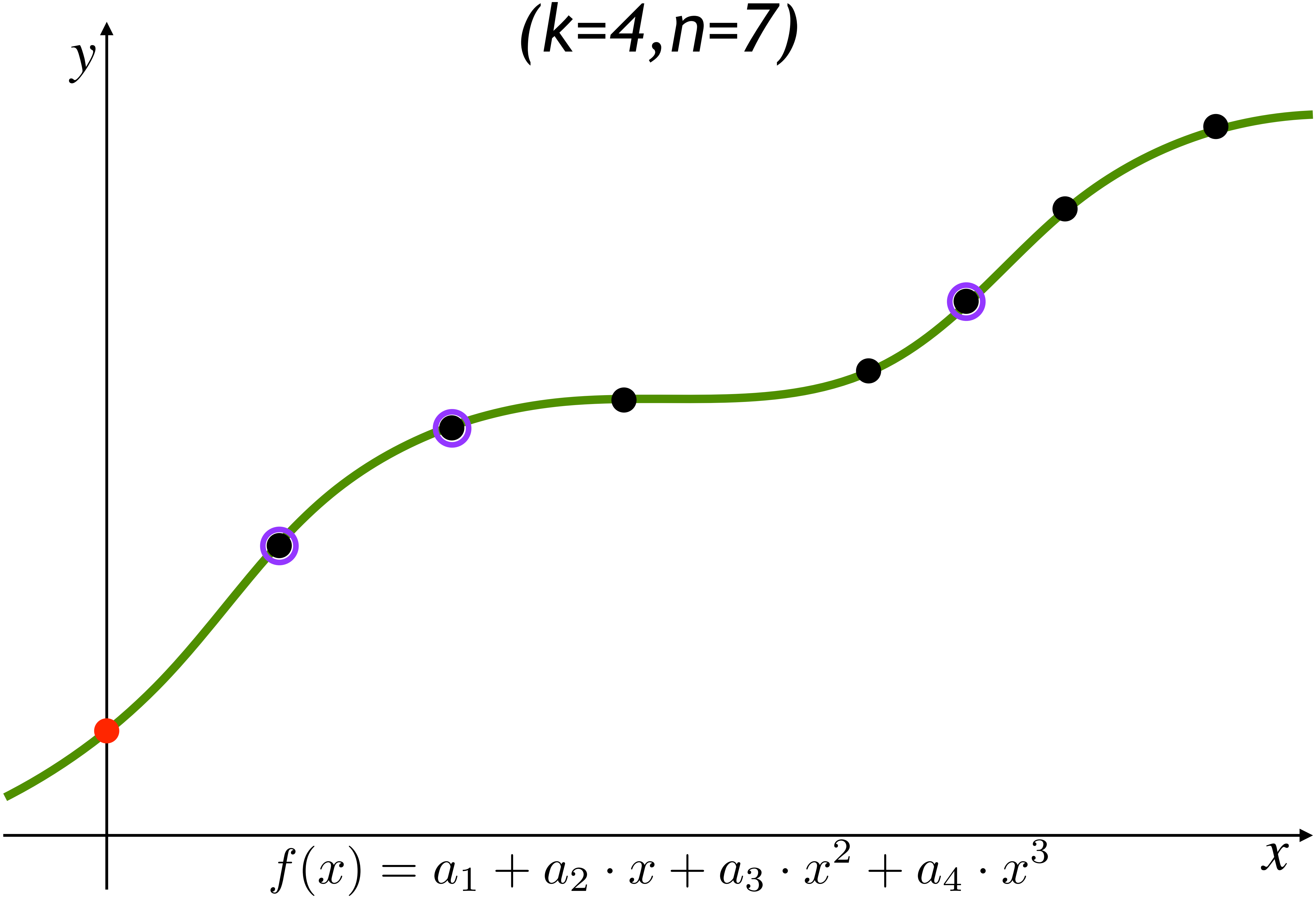
$$f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3$$

(k=4,n=7)



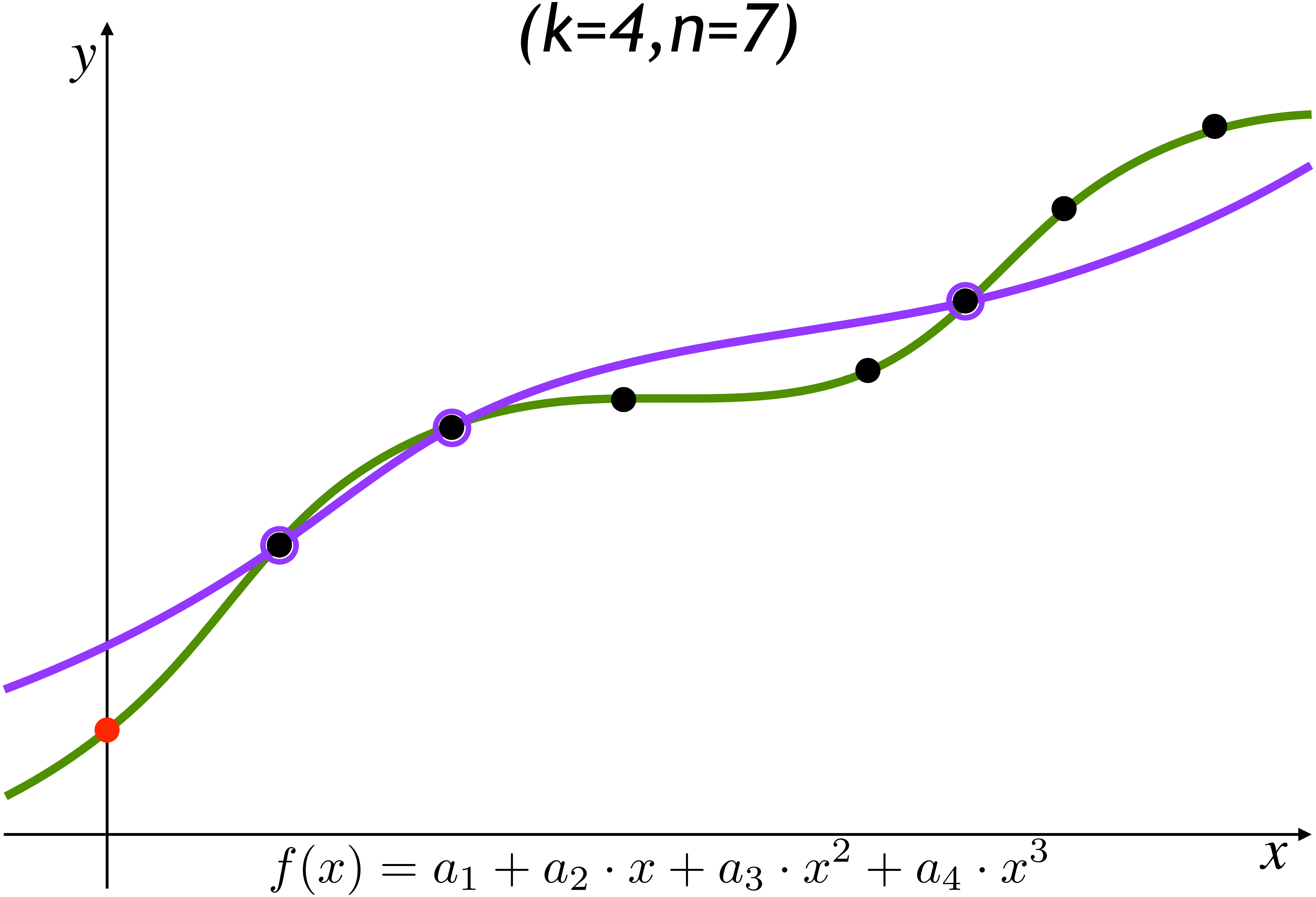
$$f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3$$

(k=4,n=7)



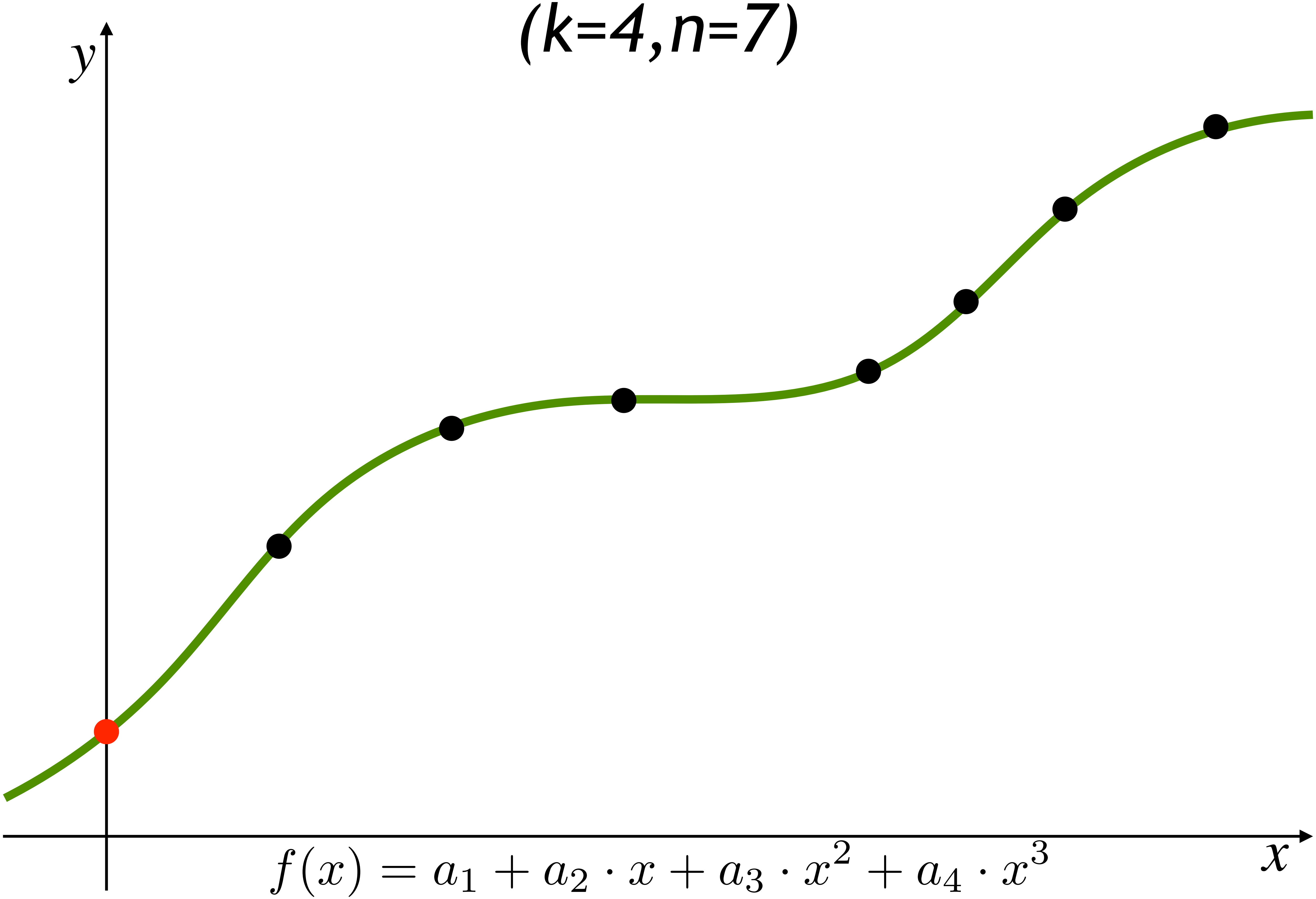
$$f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3$$

(*k*=4,*n*=7)



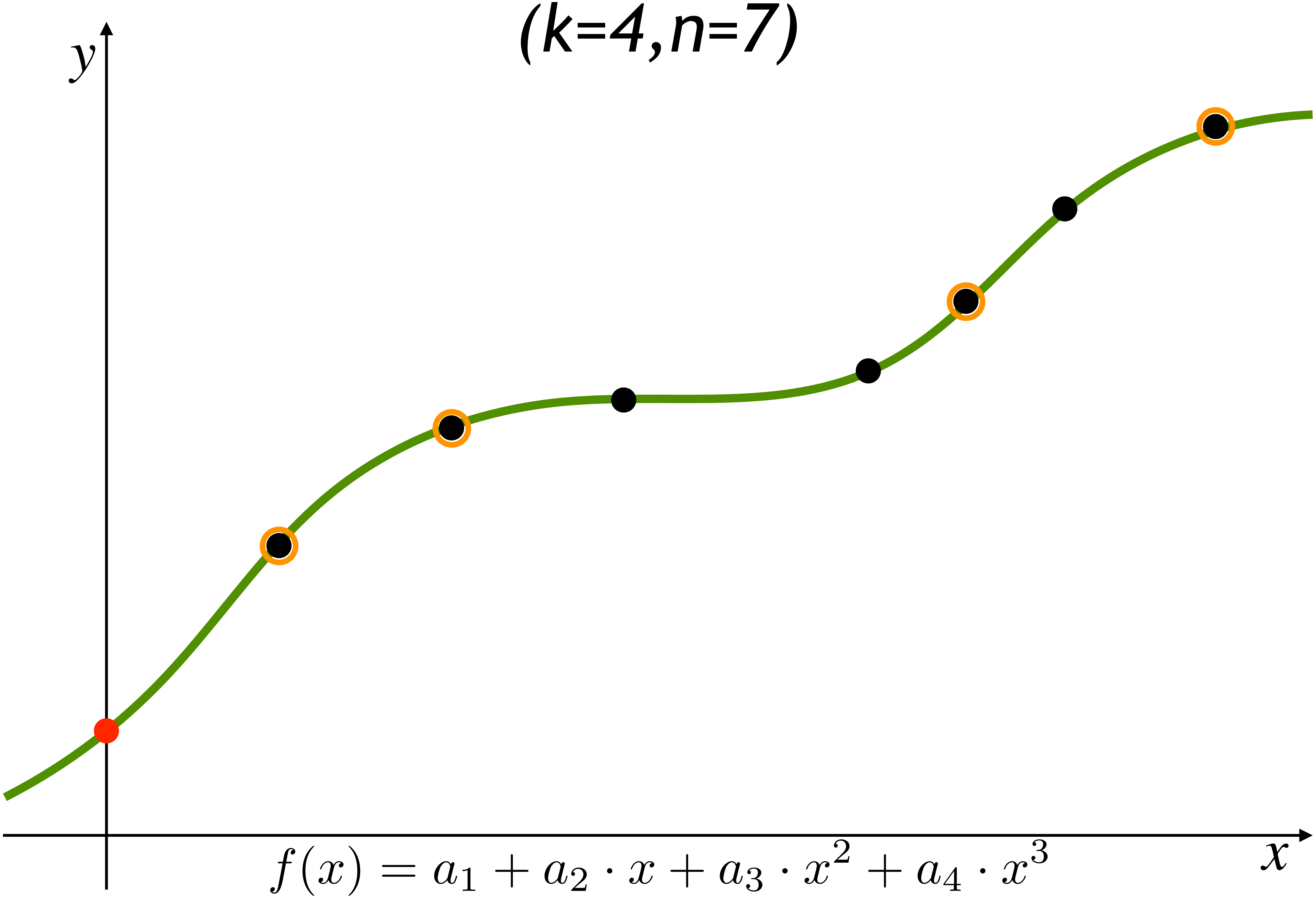
$$f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3$$

(k=4,n=7)



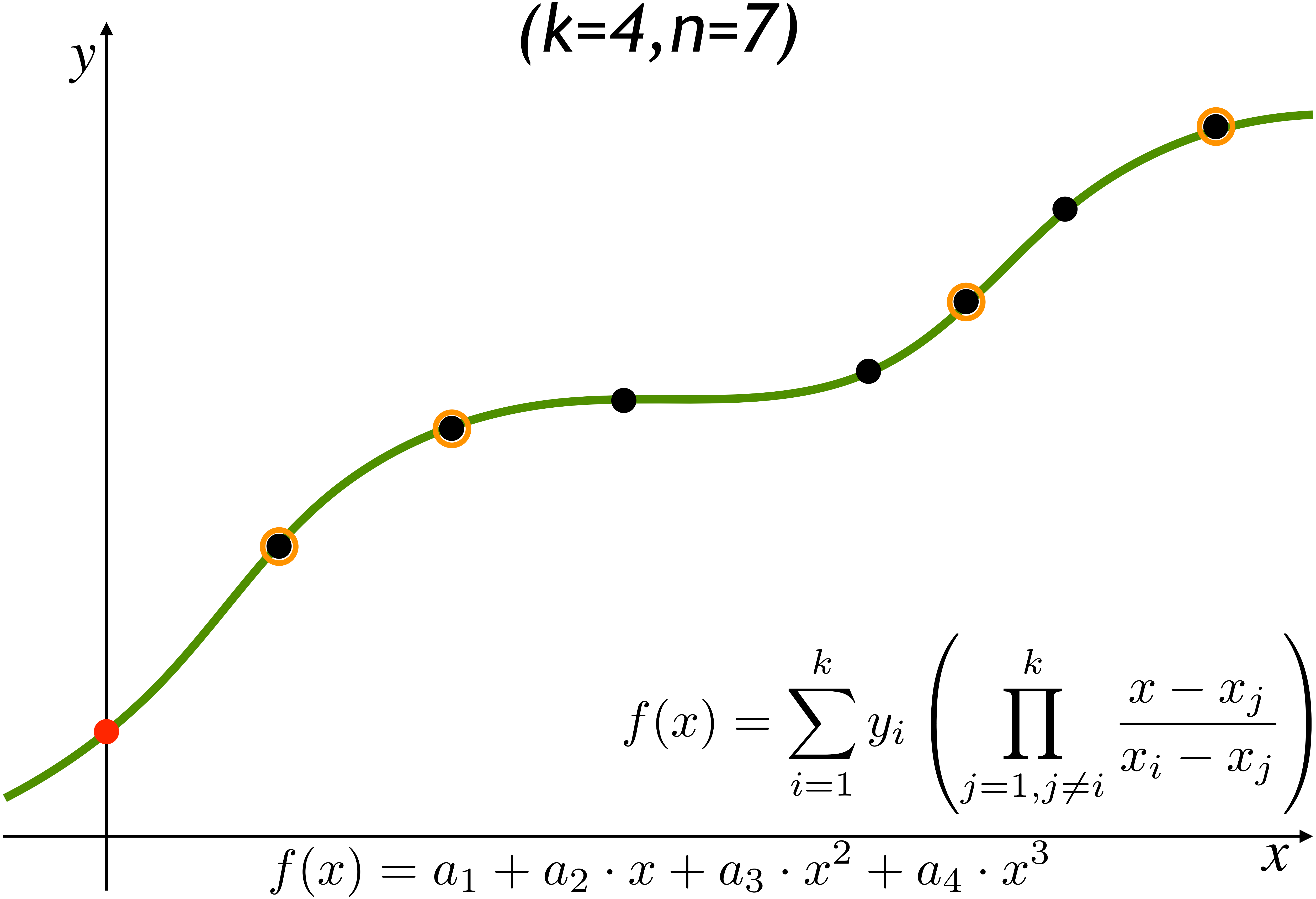
$$f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3$$

(k=4,n=7)



$$f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3$$

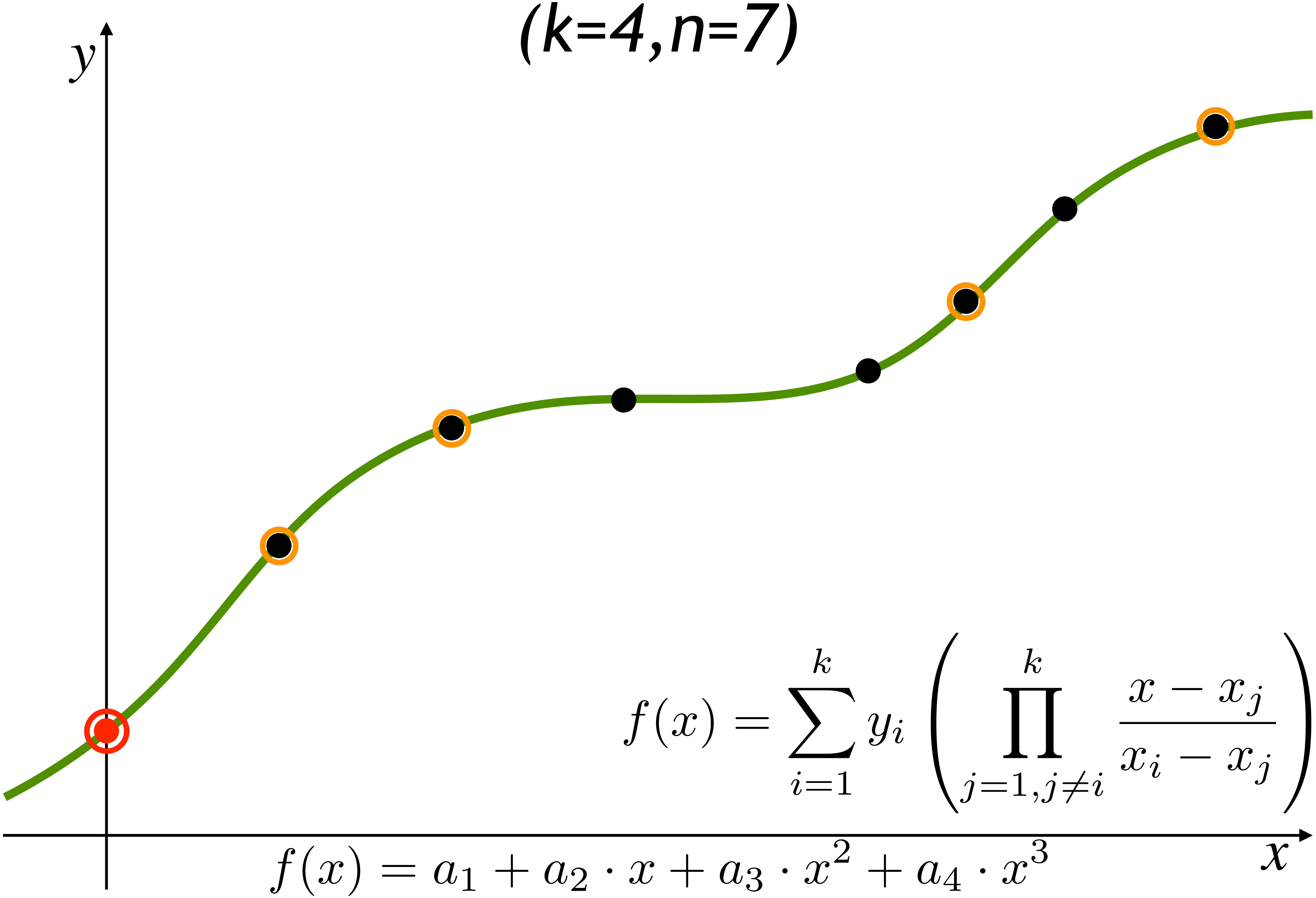
$(k{=}4, n{=}7)$



$$f(x) = \sum_{i=1}^{k} y_i \left(\prod_{j=1, j\neq i}^{k} \frac{x - x_j}{x_i - x_j} \right)$$

$$f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3$$

(k=4,n=7)



$$f(x) = \sum_{i=1}^{k} y_i \left(\prod_{j=1, j \neq i}^{k} \frac{x - x_j}{x_i - x_j} \right)$$

$$f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3$$

(k,n) threshold scheme

- $D = [x_1, \ldots, x_n]$ is a data composed of n pieces
- When at least k pieces x_i of D are known
  - D can be computed
- otherwise D remains undetermined

(k,n) threshold scheme

- $D = [x_1, \ldots, x_n]$ is a data composed of n pieces

A polynomial of degree k-1 is uniquely identified with k points

  - D can be computed
- otherwise D remains undetermined

Shamir’s (k,n) Threshold Scheme

- Let *D* be our secret (an integer), decomposed in *n* pieces
- Let *p* be a prime number *p* > max(*D*, *n*)
- Generate *k-1* random number a_i

$$\forall i \in [1; k-1] | a_i \in [0; p[$$

- Define the polynomial of degree *k-1*

$$g(x) = D + a_1 \cdot x^1 + \cdots + a_{k-1} \cdot x^{k-1}$$

- Note that *g(0)* = *D*

Shamir's (k,n) Threshold Scheme (contd.)

- Generate *n* fragments of the secret

$D_1 = g(1) \mod p$, $D_2 = g(2) \mod p$, … $D_n = g(n) \mod p$

- Distribute (x_i, D_i)
- Recompute *D* from *k* fragments (x_j, D_j) among *n* using Lagrange polynomial interpolation

$$g(0) = \sum_{i=1}^{k} D_i \left(\prod_{j=1, j \neq i}^{k} \frac{-x_j}{x_i - x_j} \right)$$

$$D \equiv g(0) \mod p$$

Example k=3, n=5

$$f(x) = \sum_{i=1}^{k} y_i \left(\prod_{j=1, j\neq i}^{k} \frac{x - x_j}{x_i - x_j} \right)$$

- p = 997
- Make 5 groups
  - group 1: (1, 547)
  - group 2: (2, 629)
  - group 3: (3, 394)
  - group 4: (4, 839)
  - group 5: (5, 967)

Example k=3, n=5

$$f(x) = \sum_{i=1}^{k} y_i \left(\prod_{j=1, j \neq i}^{k} \frac{x - x_j}{x_i - x_j} \right)$$

- p = 997
- Make 5 groups
  - group 1: (1, 547)

Collaborate with 2 other groups to compute the secret D

  - group 4: (4, 839)
  - group 5: (5, 967)

Example k=3, n =5 (contd.)

- Group 1, 3, 4

$$g(0) = 547\left(\frac{-3}{1-3}\frac{-4}{1-4}\right) + 394\left(\frac{-1}{3-1}\frac{-4}{3-4}\right) + 839\left(\frac{-1}{4-1}\frac{-3}{4-3}\right)$$

$$g(0) = 547 * 2 - 394 * 2 + 839 = 1145$$

$$g(0)\ mod\ 997 = 148$$

Example k=3, n =5 (contd.)

- To compute it, we took D = 148, *p* = 997 a prime number, and the polynomial

  p=997 (prime), a_1=59 (random), a_2=340(random)

  $g(x)=148 + 59x + 340x^2$

- Such that

  $D_1 = g(1) \bmod 997 = 547$

  $D_2 = g(2) \bmod 997 = 1626 \bmod 997 = 629$

  $D_3 = g(3) \bmod 997 = 3385 \bmod 997 = 394$

  $D_4 = g(4) \bmod 997 = 5824 \bmod 997 = 839$

  $D_5 = g(5) \bmod 997 = 8943 \bmod 997 = 967$

Shamir’s (k,n) Threshold Scheme (contd.)

- The size of each fragment does not exceeds the size of the secret
  - as long as *p* is chosen of the same order as the secret
- Possible to generate new fragments at any time, without altering the others
- Possible to construct hierarchies by attributing more or less fragments
  - the boss has k fragments, the subaltern has k/2, …
- No assumption as opposed to cryptographic functions

Anonymity

- Alice wants to send a message to Bob
  - Communications are unsecured
  - Nobody can know who is the sender (not even Bob)
  - Nobody can know who is the receiver
  - Nobody else than Bob can retrieve the message

Mix

- Objectives of a mix
  - Hide correspondences between incoming and outgoing messages
  - Not possible to map a source and an outgoing message (apart for the mix)
  - No possible to map a receiver and an incoming message (apart for the mix)

Mix (contd.)

- If the mix cannot be fully trusted, use a cascade of mixes
- It works as long as untrusted mixes do not collaborate all together

Chaum-net

- Allow to send a sealed message via a cascade of mixes
- In an overlay, each participant has a private/public key pair
- Alice randomly choses a few of them (e.g., 3) to be mixes
- Alice recursively encrypt the message with the public key of each mixes she selected

Chaum-net

- Allow to send a sealed message via a cascade of mixes
- In an overlay, each participant has a private/public key pair
- Alice randomly choses a few of them (e.g., 3) to be mixes
- Alice recursively encrypt the message with the public key of each mixes she selected

Chaum-net example

Alice

Bob

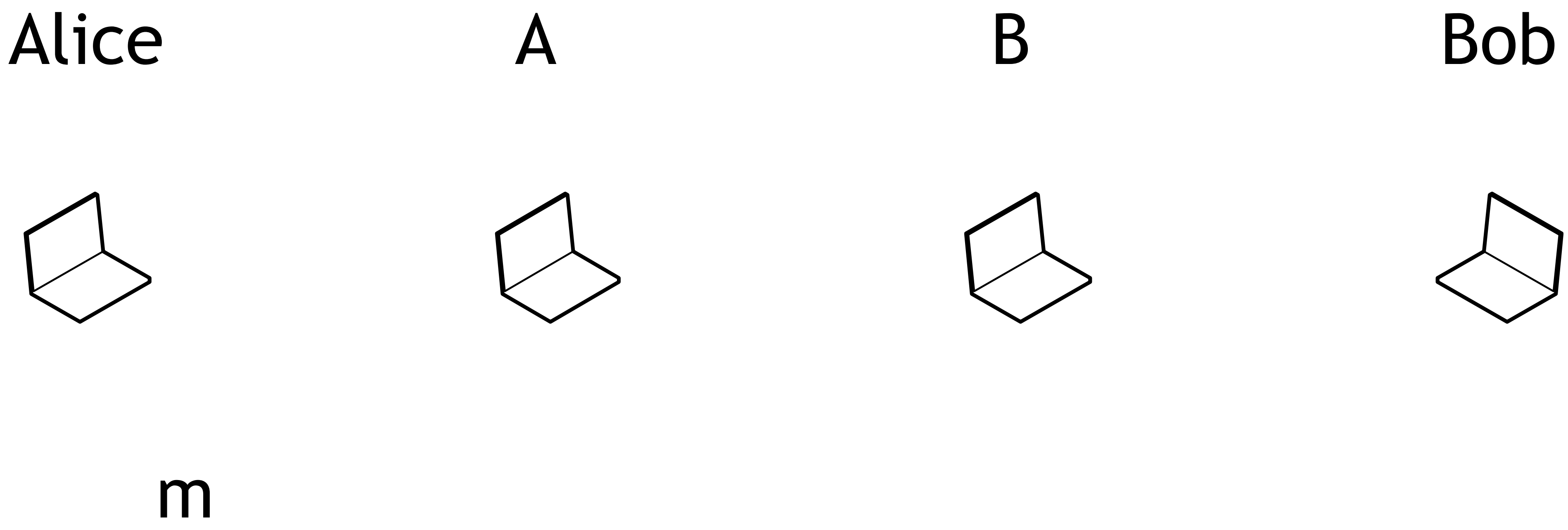
Chaum-net example

Alice

Bob

m

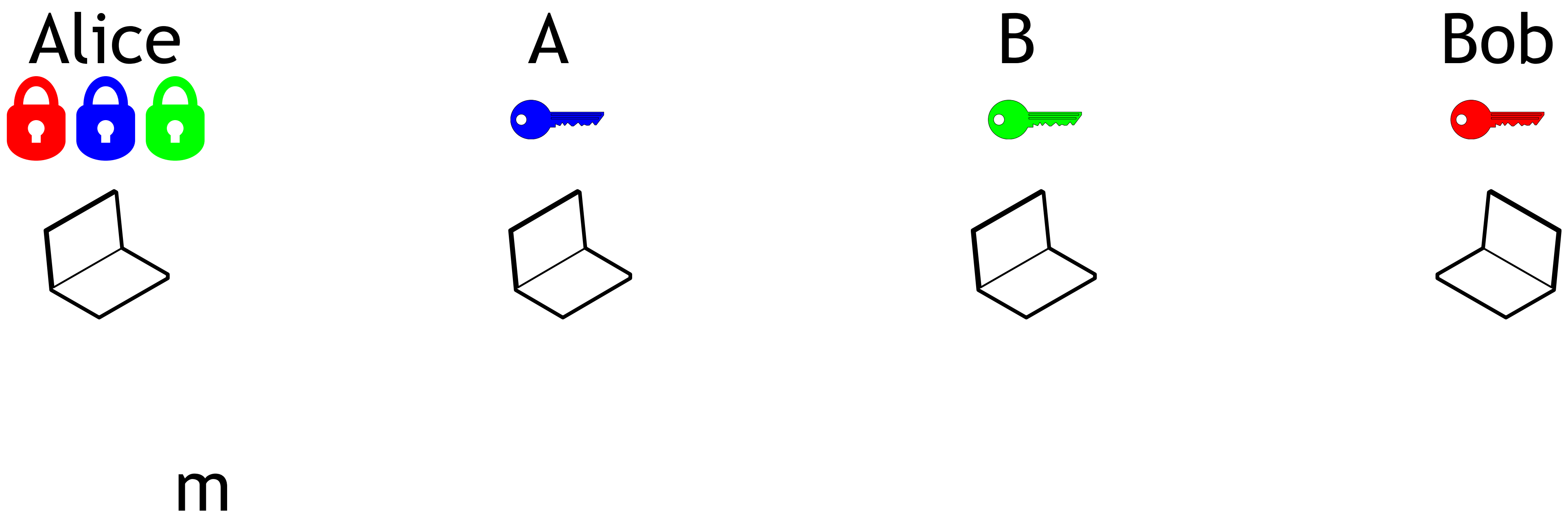
Chaum-net example



Alice A B Bob

m

Chaum-net example



Alice

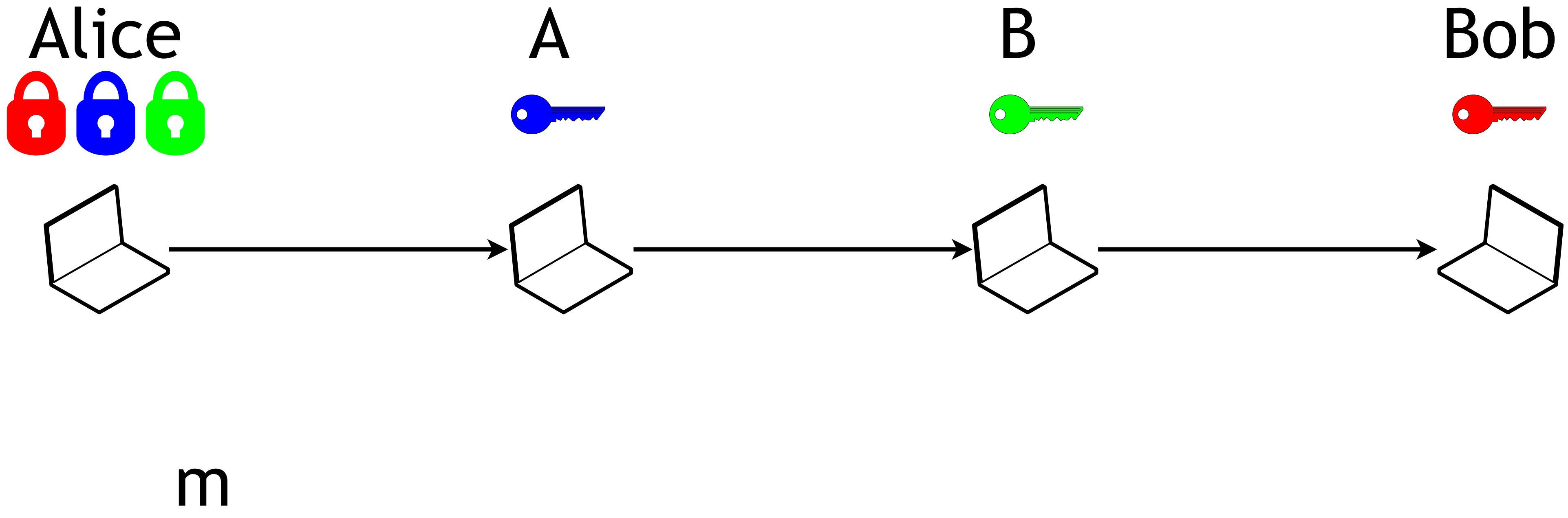
A

B

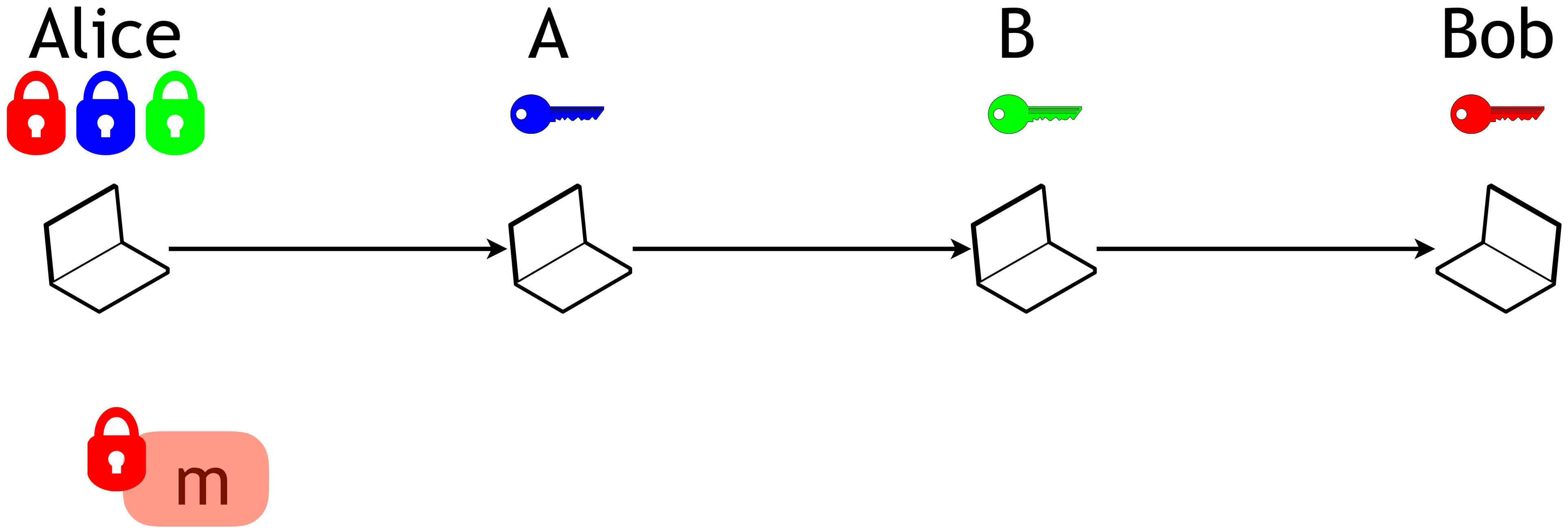
Bob

m

Chaum-net example

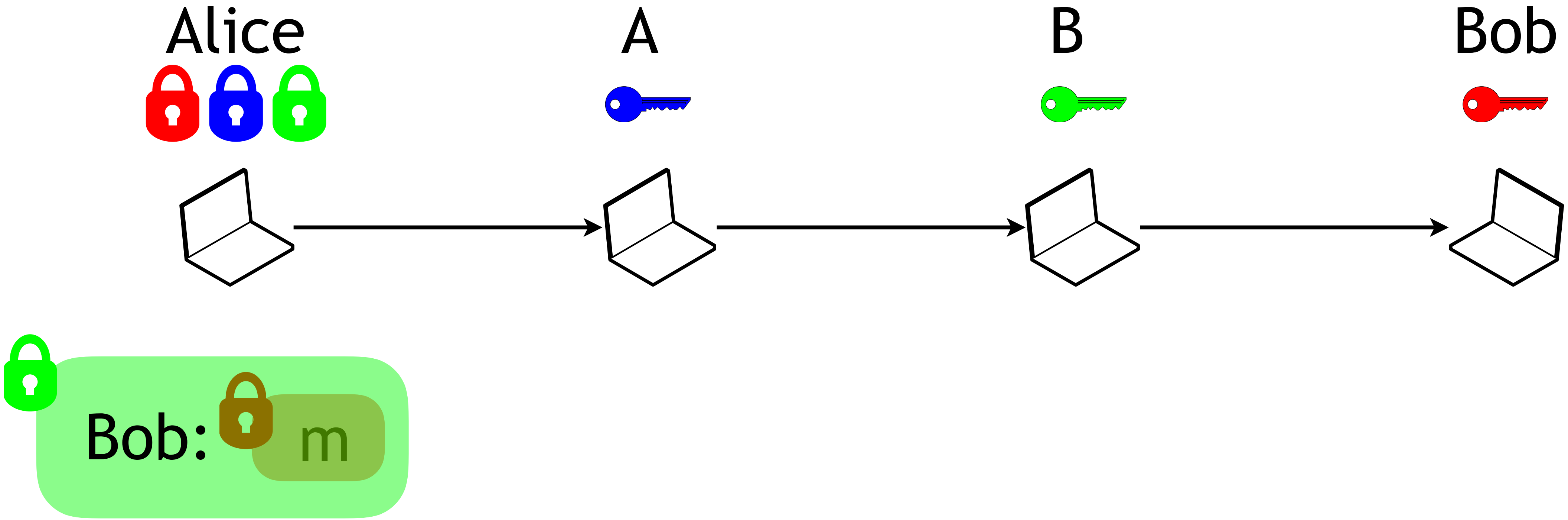


Chaum-net example



$K_{Bob}(R_0, m)$

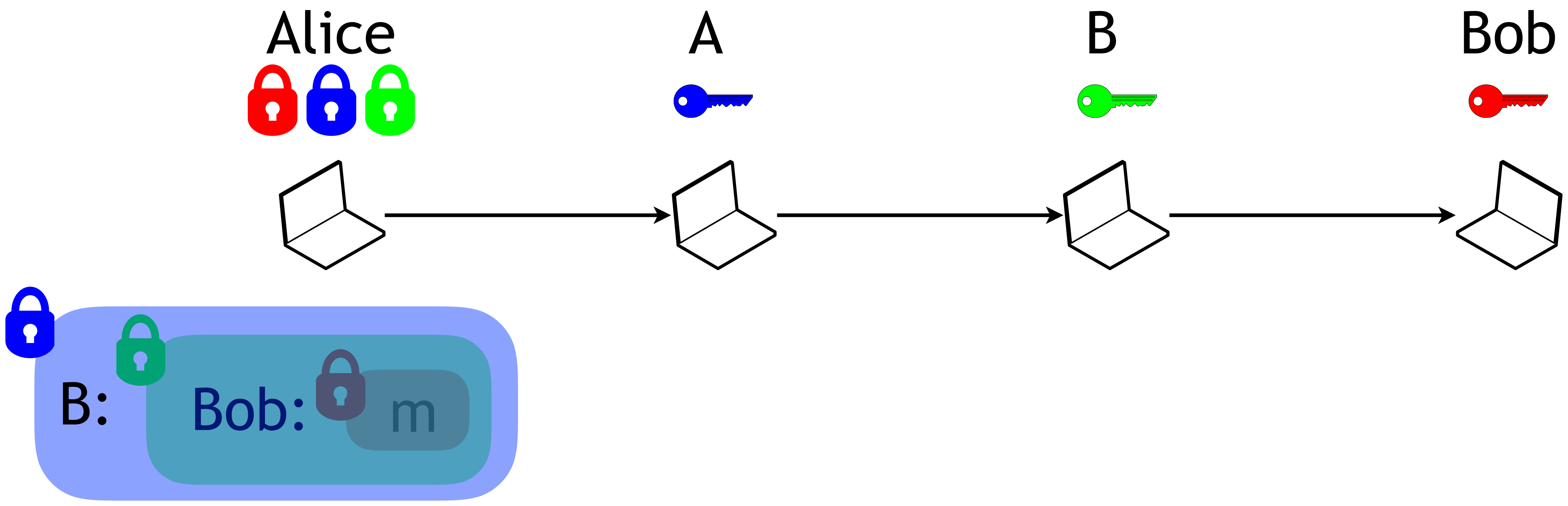
Chaum-net example



$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$

Chaum-net example

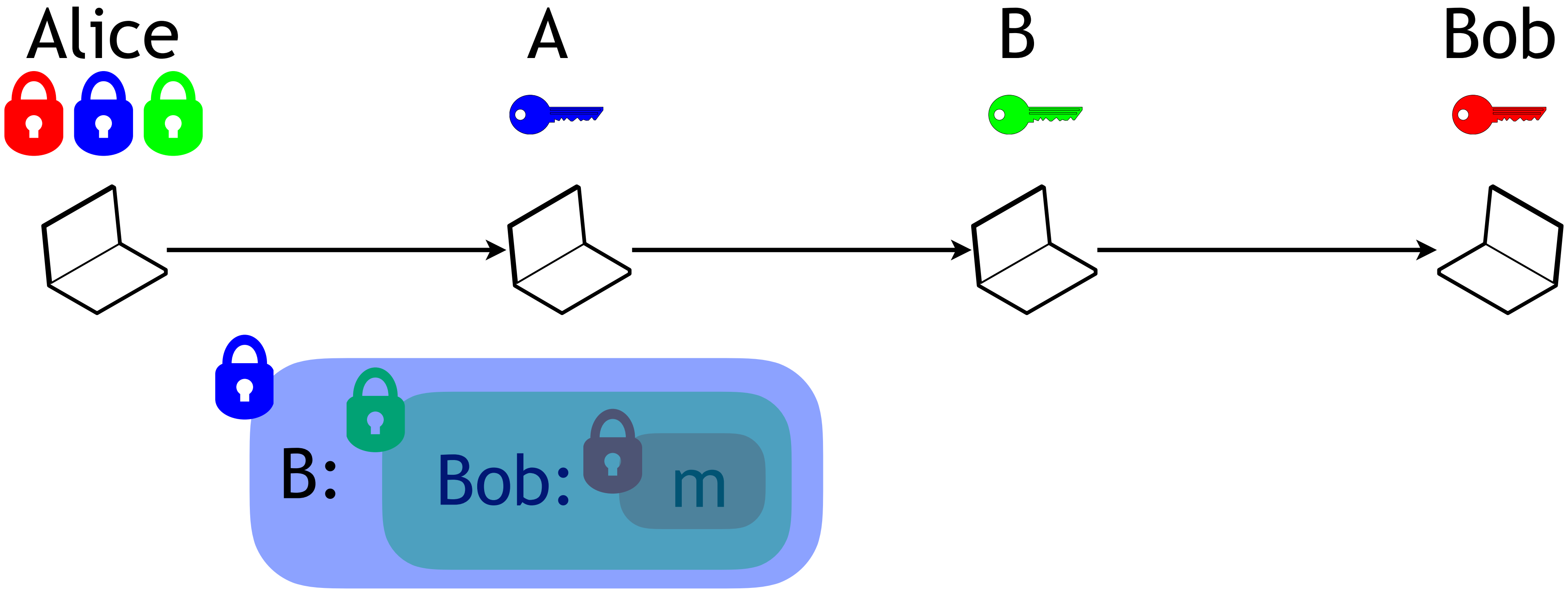


$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$

$K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))$

Chaum-net example

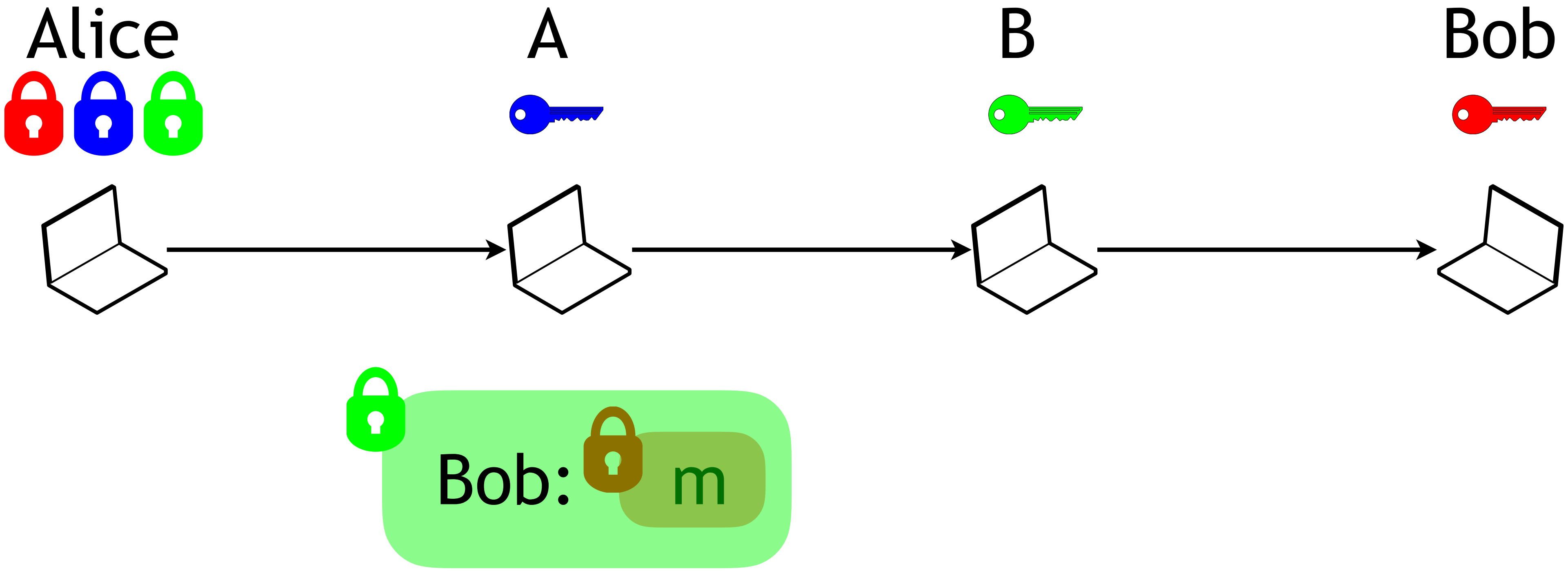


$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$

$K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))$

Chaum-net example

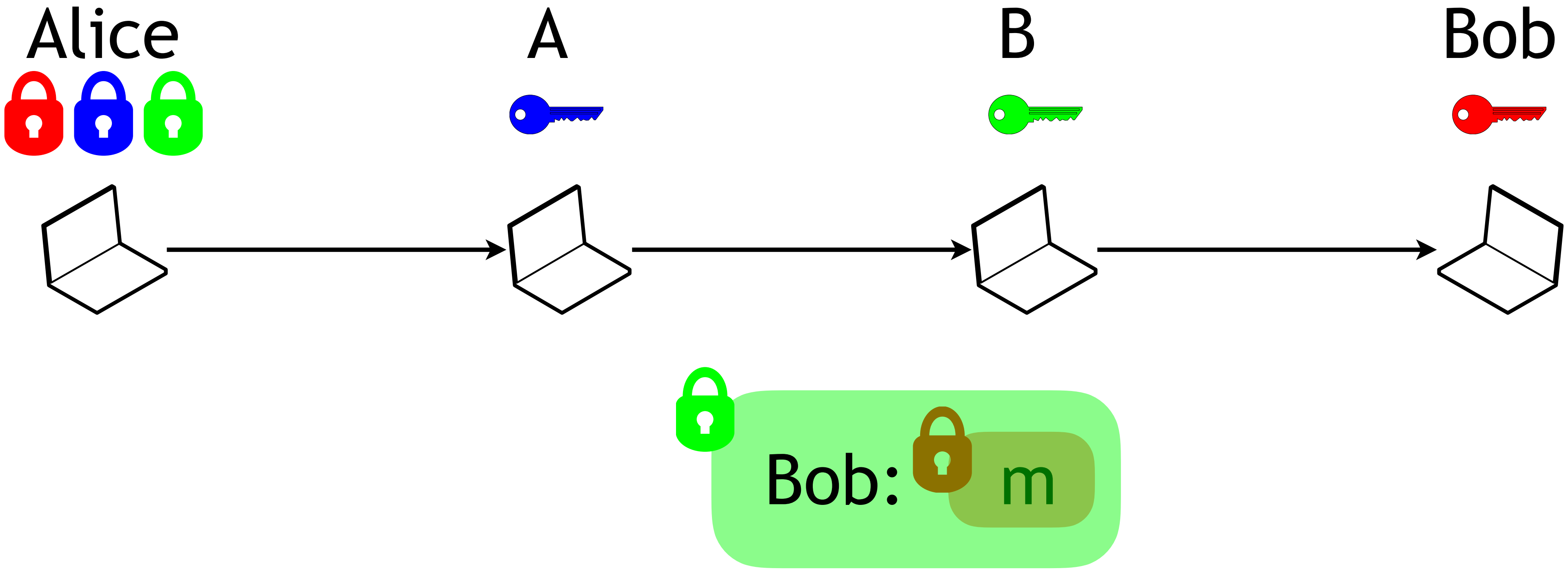


$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$

$K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))$

Chaum-net example

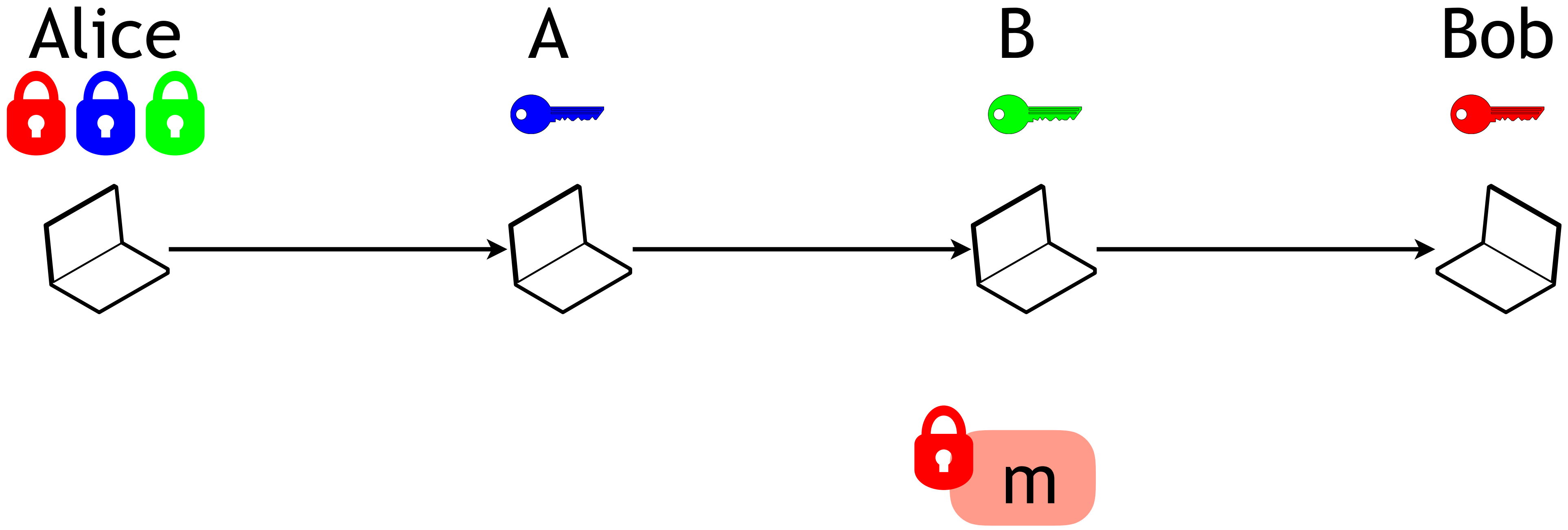


$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$

$K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))$

Chaum-net example

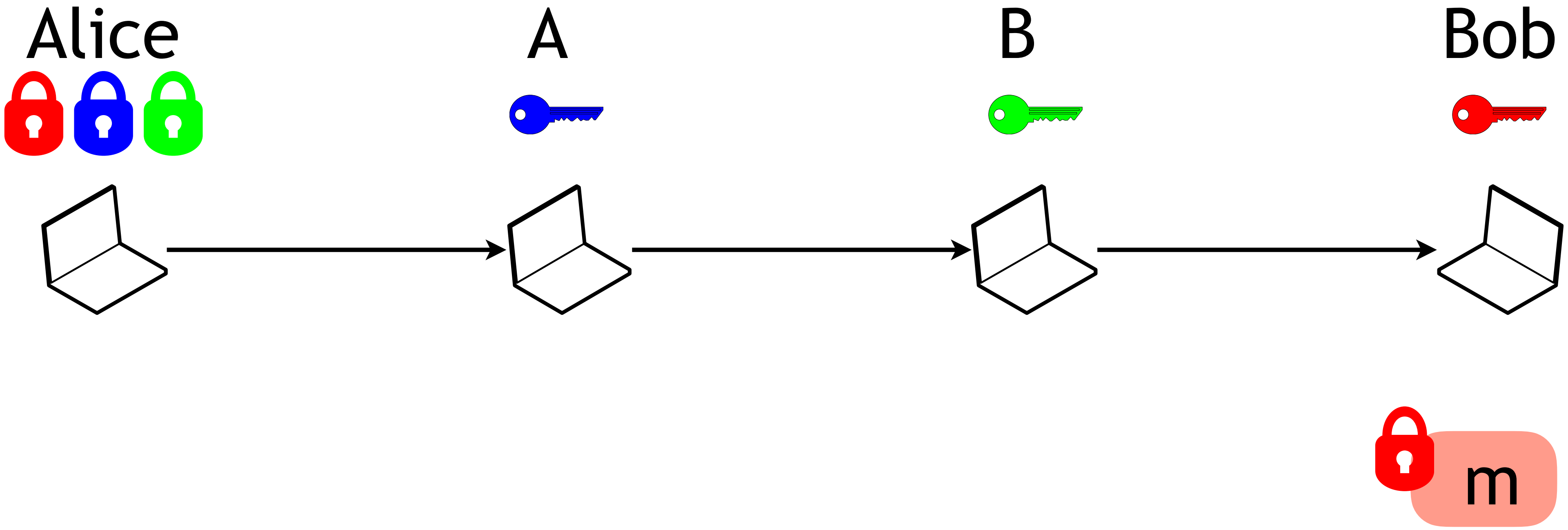


$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$

$K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))$

Chaum-net example

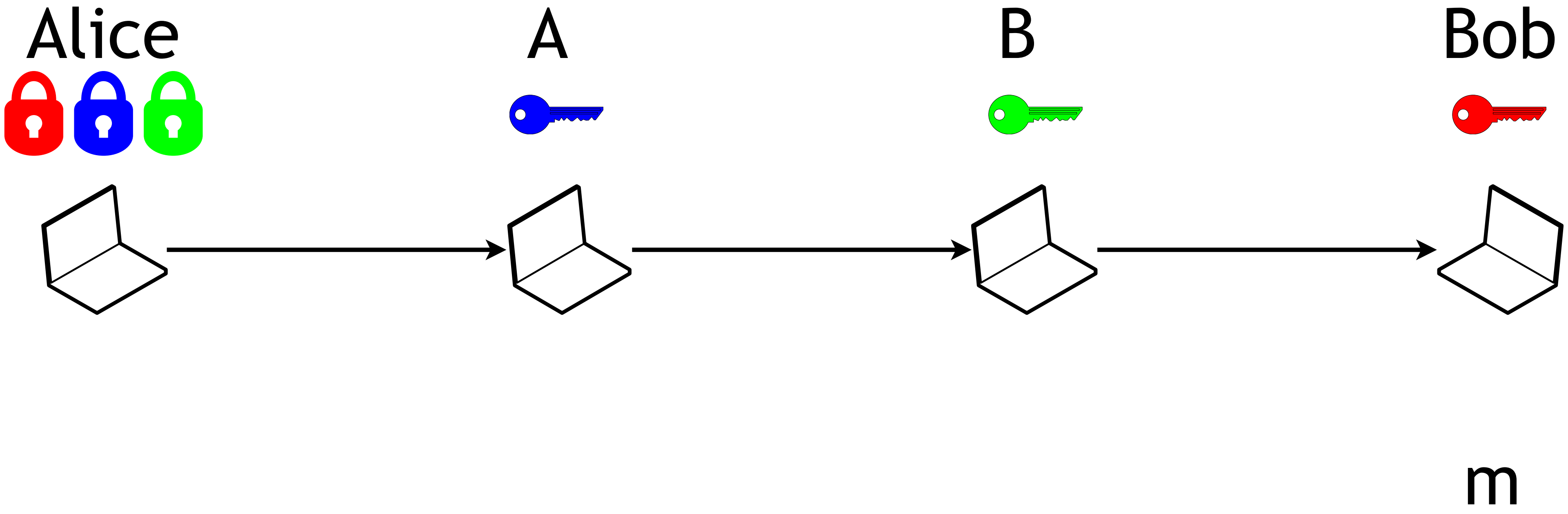


$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$

$K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))$

Chaum-net example

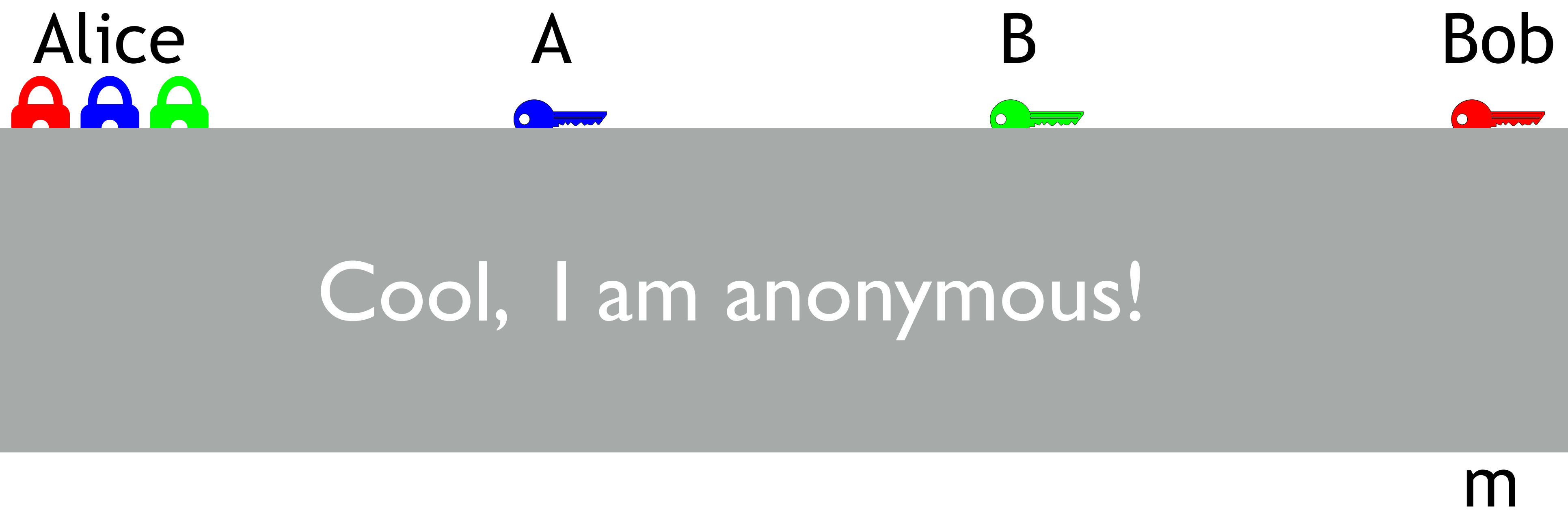


$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$

$K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))$

Chaum-net example



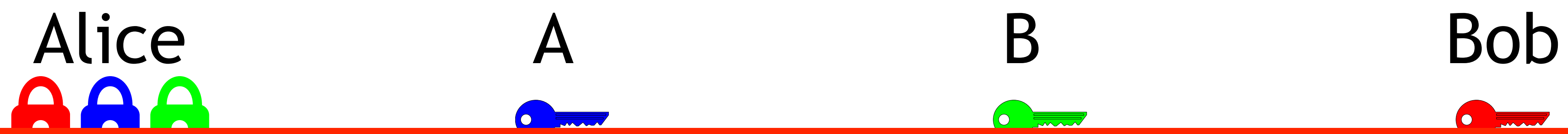
Cool, I am anonymous!

$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$

$K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))$

Chaum-net example



Are you sure?

m

$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$

$K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))$

Social behavior

"If you have something that you don't want anyone to know, maybe you shouldn't be doing it in the first place."

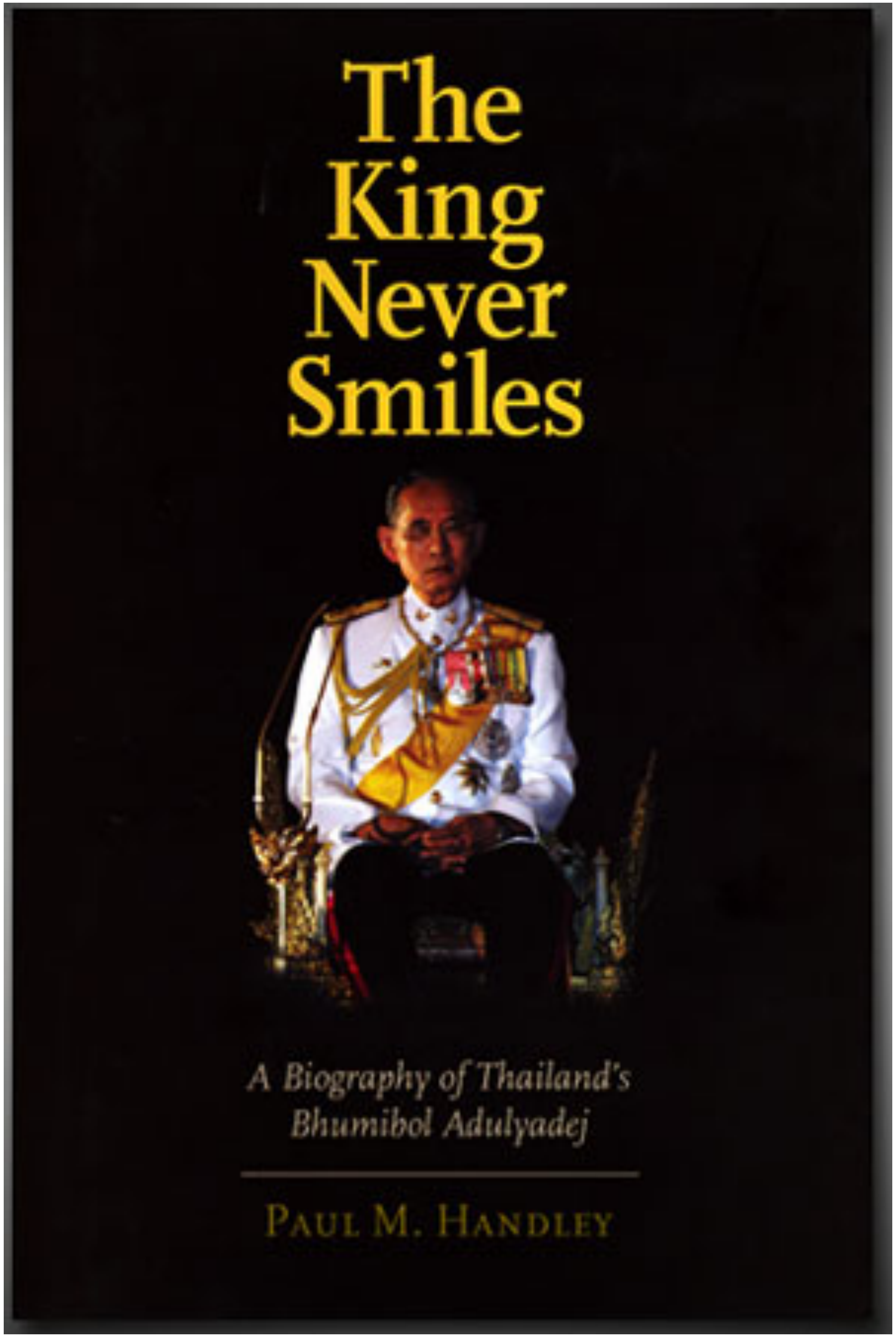
"If you have something that you don't want anyone to know, maybe you shouldn't be doing it in the first place."

Eric Schmidt, directeur général de Google, 2009

Je n’ai rien à cacher!

Je n’ai rien à cacher!

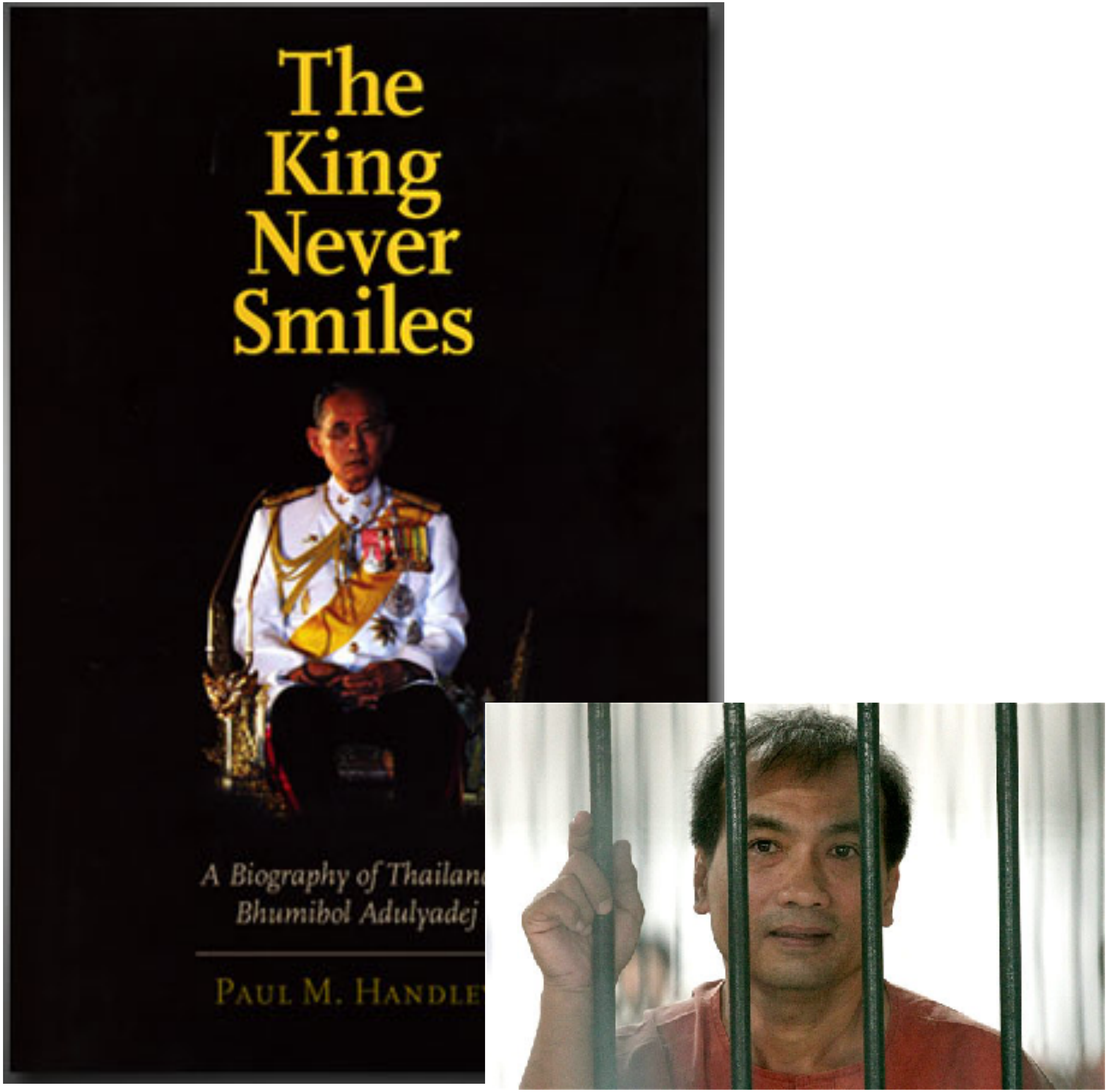
Les définitions de lois et moralité ne sont pas universelles

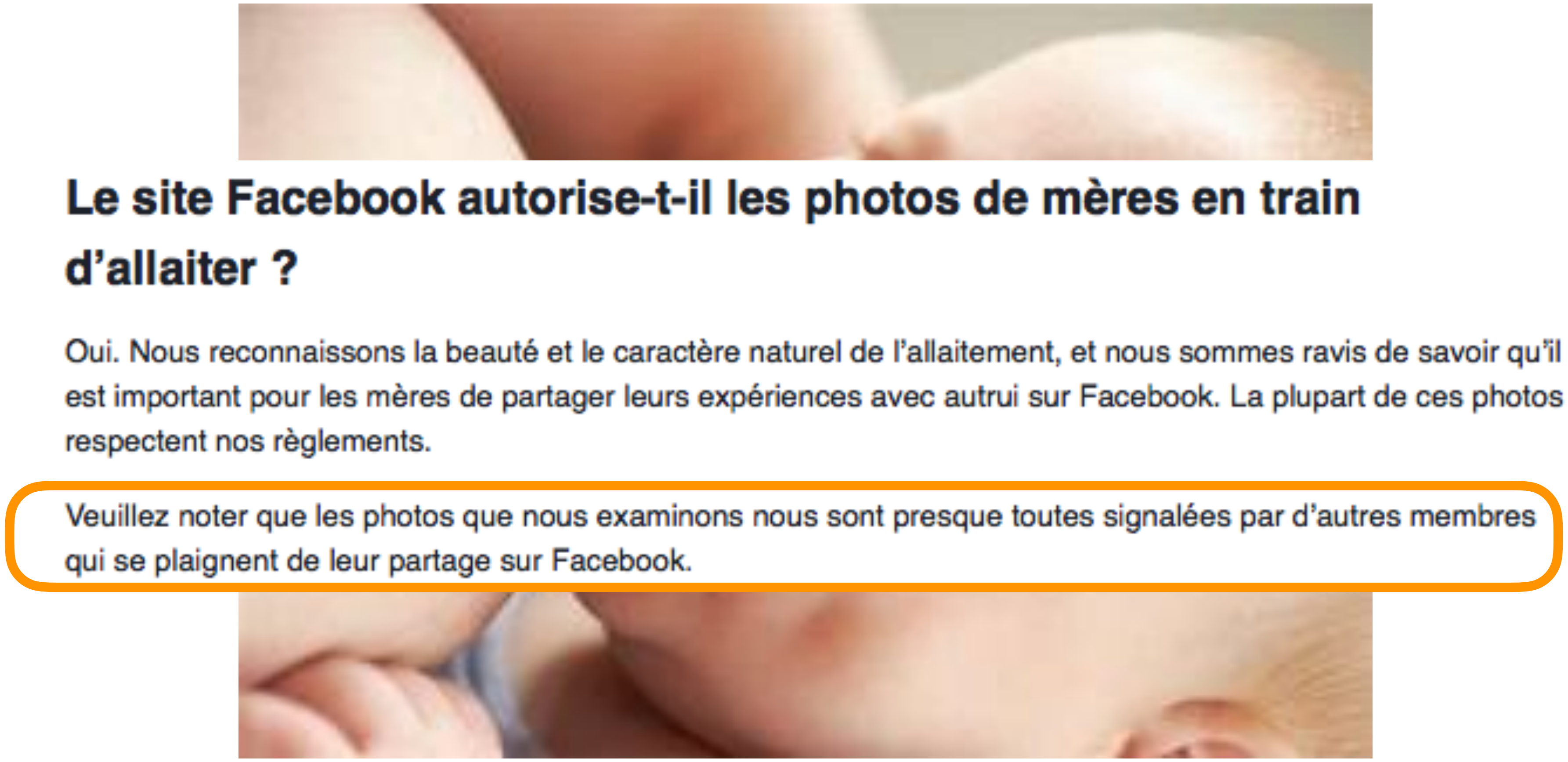


The King Never Smiles

A Biography of Thailand's Bhumibol Adulyadej

PAUL M. HANDLEY

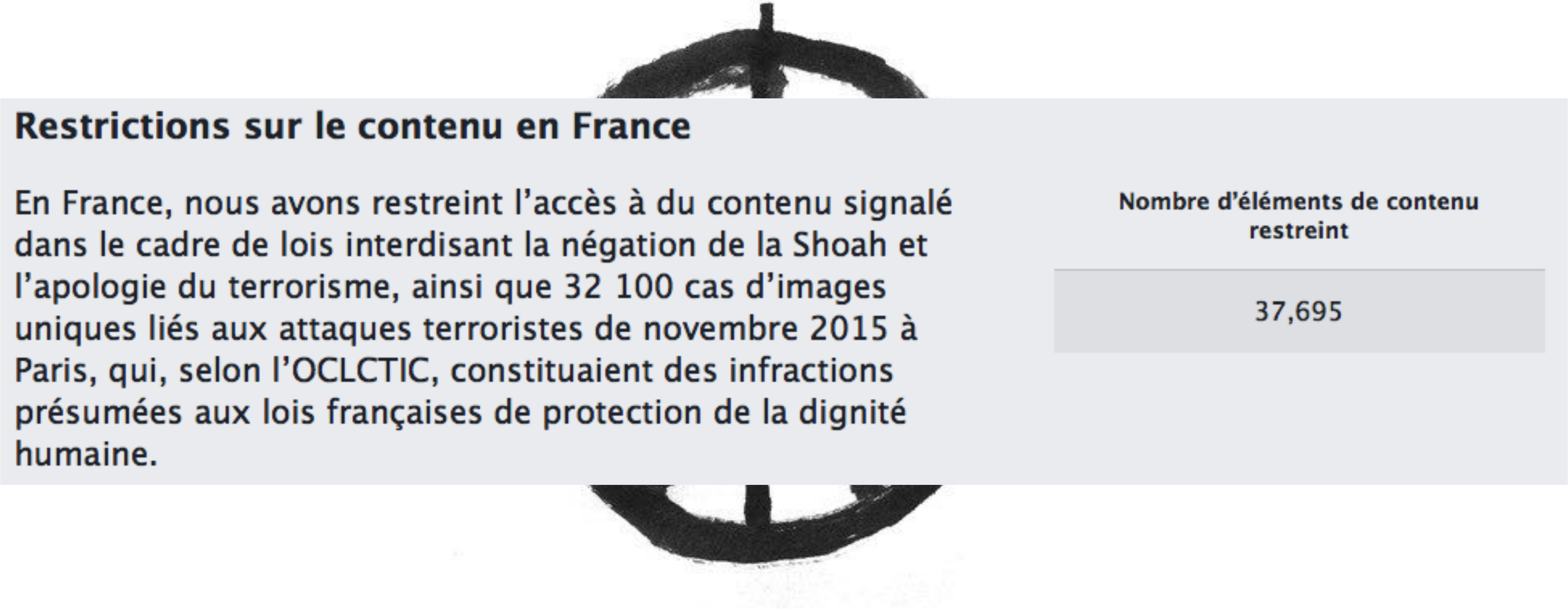

The King Never Smiles
A Biography of Thailan
Bhumibol Adulyadej
PAUL M. HANDLE



Le site Facebook autorise-t-il les photos de mères en train d’allaiter ?

Oui. Nous reconnaissons la beauté et le caractère naturel de l’allaitement, et nous sommes ravis de savoir qu’il est important pour les mères de partager leurs expériences avec autrui sur Facebook. La plupart de ces photos respectent nos règlements.

Veuillez noter que les photos que nous examinons nous sont presque toutes signalées par d’autres membres qui se plaignent de leur partage sur Facebook.



Restrictions sur le contenu en France

En France, nous avons restreint l'accès à du contenu signalé dans le cadre de lois interdisant la négation de la Shoah et l'apologie du terrorisme, ainsi que 32 100 cas d'images uniques liés aux attaques terroristes de novembre 2015 à Paris, qui, selon l'OCLCTIC, constituaient des infractions présumées aux lois françaises de protection de la dignité humaine.

Nombre d'éléments de contenu restreint
37,695

Je suis invisible sur Internet

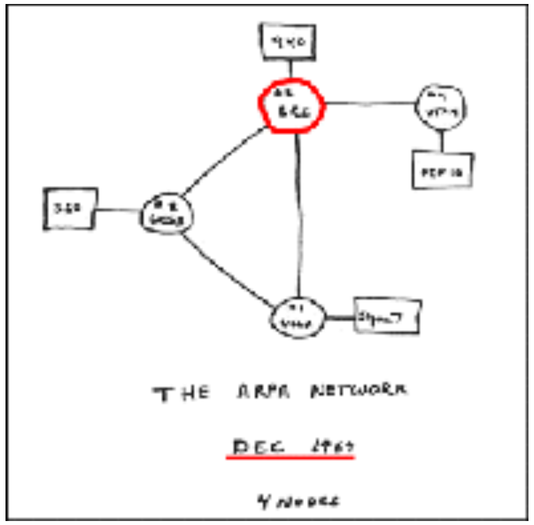
Je suis invisible sur Internet

Mais je l’utilise tout le temps et partout

Je suis invisible sur Internet

Mais je l'utilise tout le temps et partout

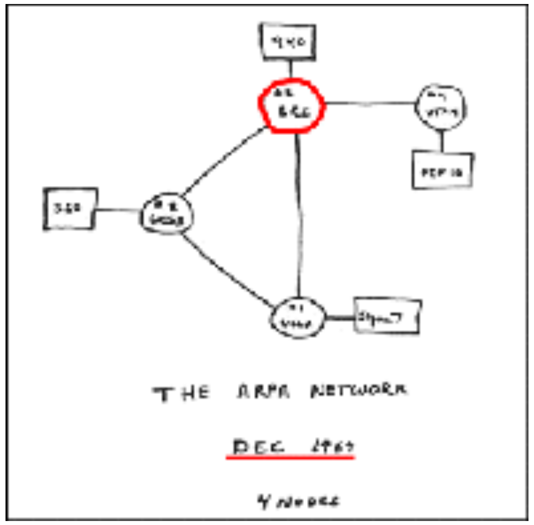
L’Internet a beaucoup changé



[ARPANET logic map,1969]

L’Internet a beaucoup changé

de 4 à plus 1 milliard de terminaux



[ARPANET logic map,1969]

En principe l’Internet est décentralisé

En principe l’Internet est décentralisé

En pratique il est contrôlé par quelques géants…

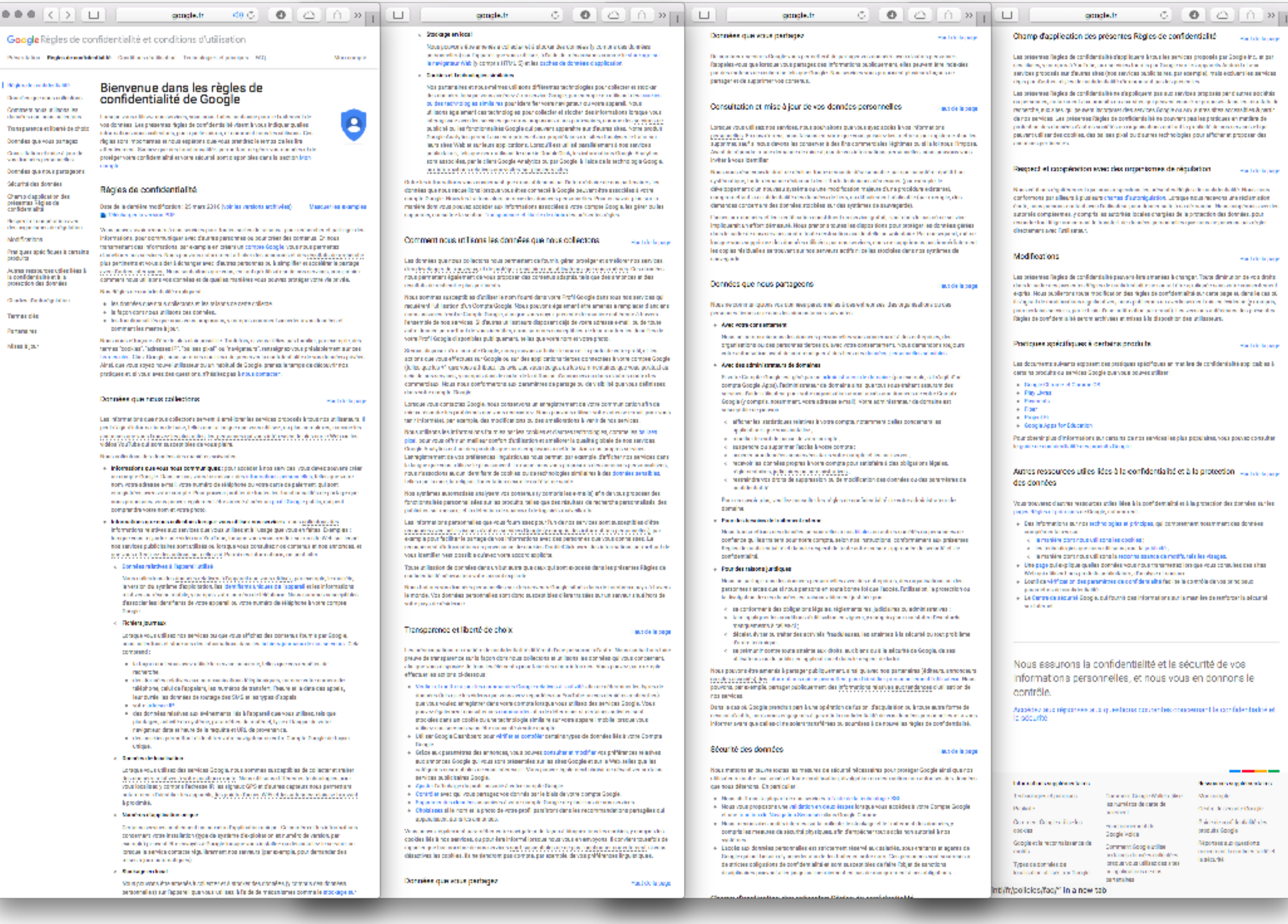
En principe l’Internet est décentralisé

En pratique il est contrôlé par quelques géants…

… chez qui il faut s’enregistrer

… chez qui il faut s’enregistrer

Cliquez ici pour accepter



google.fr

Google Règles de confidentialité et conditions d'utilisation

Bienvenue dans les règles de confidentialité de Google

Règles de confidentialité

Date de la dernière modification : 25 mars 2016 (voir les versions archivées) Masquer les exemples

Données que nous collectons

Comment nous utilisons les données que nous collectons

Transparence et liberté de choix

Données que vous partagez

Consultation et mise à jour de vos données personnelles

Données que nous partageons

Sécurité des données

Champ d'application des présentes Règles de confidentialité

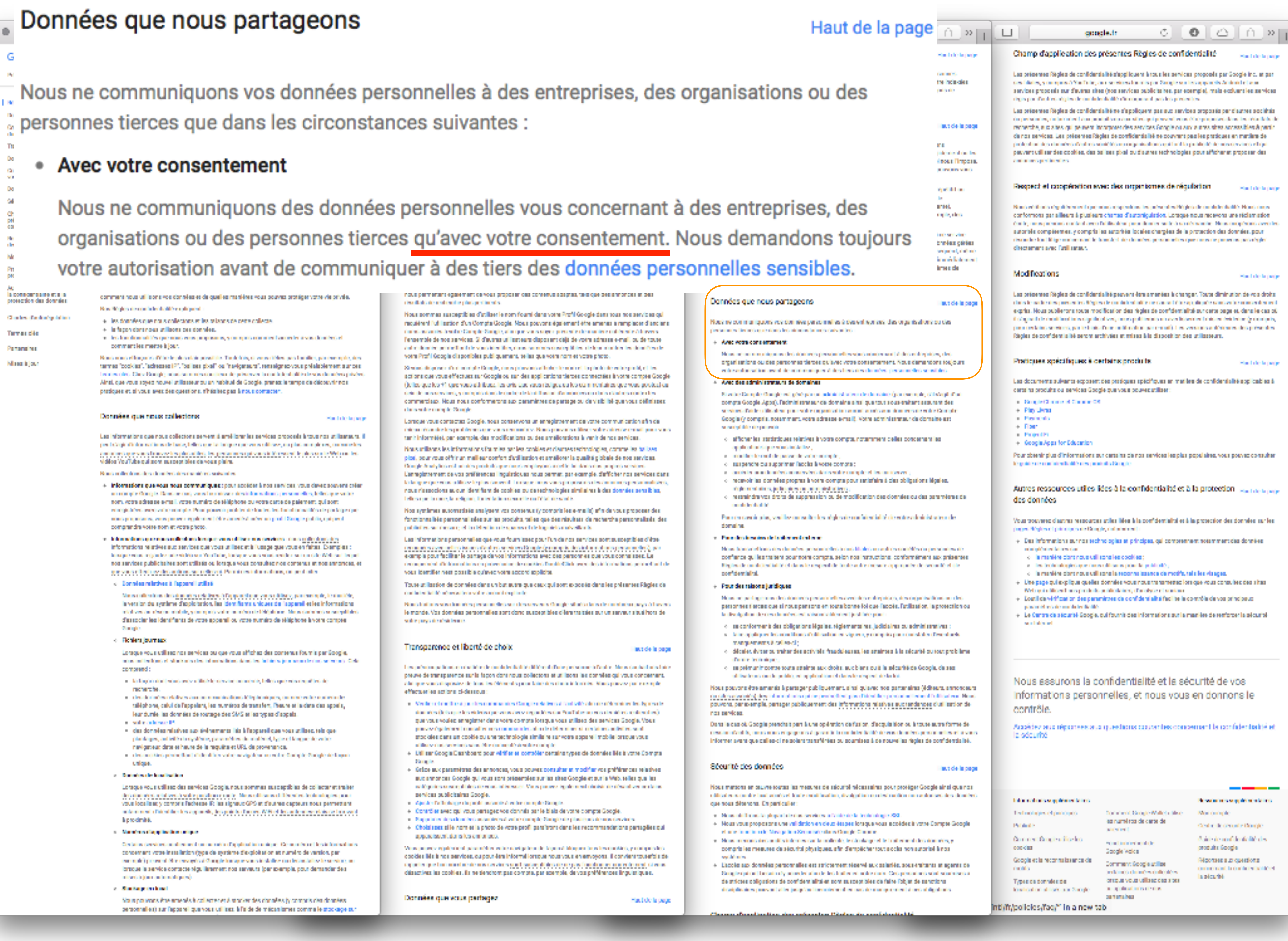
Respect et coopération avec des organismes de régulation

Modifications

Pratiques spécifiques à certains produits

Autres ressources utiles liées à la confidentialité et à la protection des données

Nous assurons la confidentialité et la sécurité de vos informations personnelles, et nous vous en donnons le contrôle.



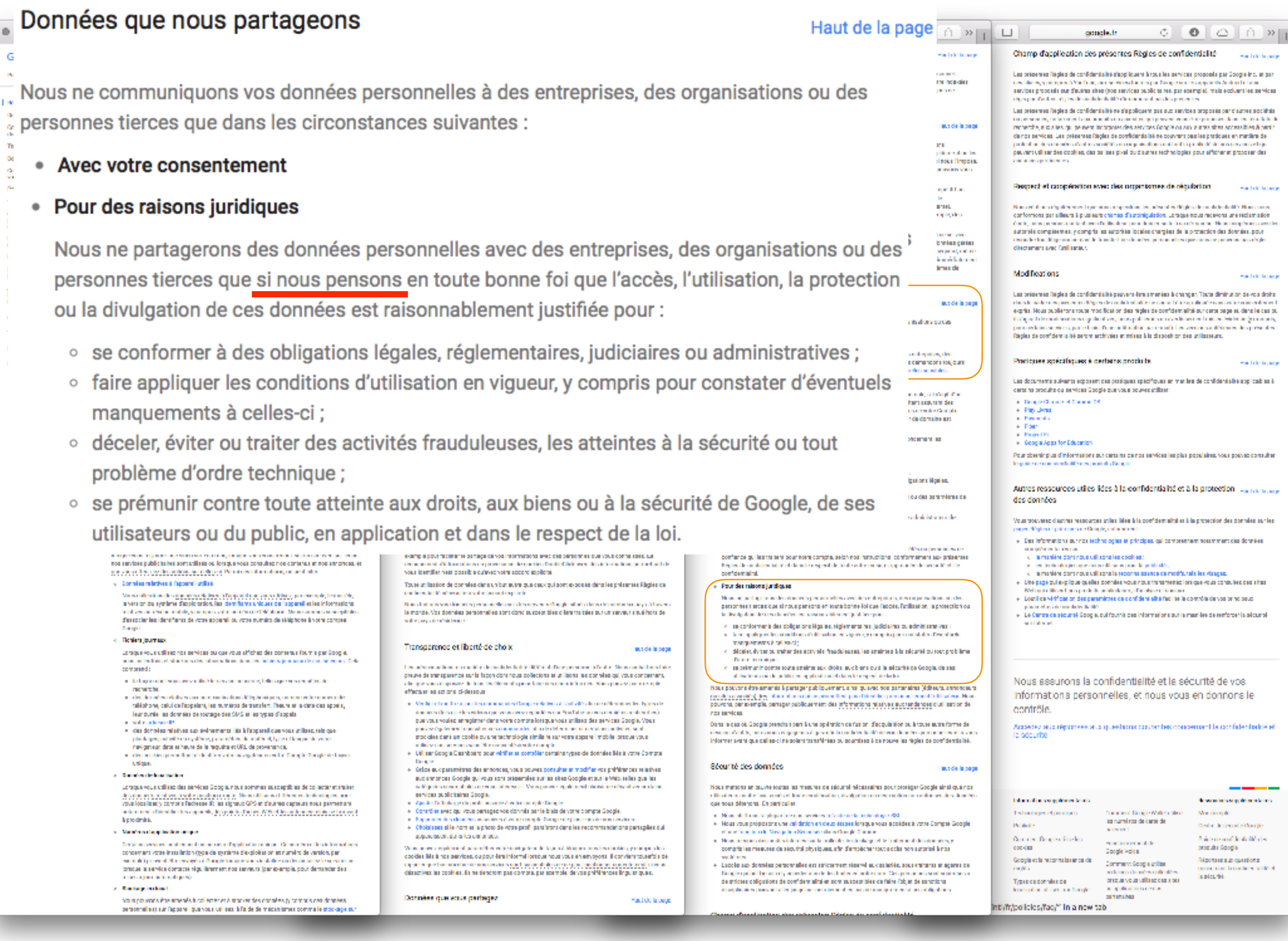
Données que nous partageons

Haut de la page

Nous ne communiquons vos données personnelles à des entreprises, des organisations ou des personnes tierces que dans les circonstances suivantes :

- **Avec votre consentement**

  Nous ne communiquons des données personnelles vous concernant à des entreprises, des organisations ou des personnes tierces qu'avec votre consentement. Nous demandons toujours votre autorisation avant de communiquer à des tiers des données personnelles sensibles.



Données que nous partageons

Haut de la page

Nous ne communiquons vos données personnelles à des entreprises, des organisations ou des personnes tierces que dans les circonstances suivantes :

- **Avec votre consentement**
- **Pour des raisons juridiques**

  Nous ne partagerons des données personnelles avec des entreprises, des organisations ou des personnes tierces que si nous pensons en toute bonne foi que l'accès, l'utilisation, la protection ou la divulgation de ces données est raisonnablement justifiée pour :

  - se conformer à des obligations légales, réglementaires, judiciaires ou administratives ;
  - faire appliquer les conditions d'utilisation en vigueur, y compris pour constater d'éventuels manquements à celles-ci ;
  - déceler, éviter ou traiter des activités frauduleuses, les atteintes à la sécurité ou tout problème d'ordre technique ;
  - se prémunir contre toute atteinte aux droits, aux biens ou à la sécurité de Google, de ses utilisateurs ou du public, en application et dans le respect de la loi.

google.fr
Champ d'application des présentes Règles de confidentialité
Respect et coopération avec des organismes de régulation
Modifications
Pratiques spécifiques à certains produits
Autres ressources utiles liées à la confidentialité et à la protection des données
Nous assurons la confidentialité et la sécurité de vos informations personnelles, et nous vous en donnons le contrôle.
Transparence et liberté de choix
Données que nous partageons
Pour des raisons juridiques
Sécurité des données

… et qui son intégrés à tous les sites

… et qui son intégrés à tous les sites

Cliquez ici pour partager

Je n’ai pas de compte

Je n’ai pas de compte

Je me déconnecte

Risque pour votre vie privée

Risque pour votre vie privée

Je leur fait confiance

Qui utilise Skype?

Qui utilise Skype?

Logiciel de téléphonie par Internet composé

- d’un annuaire téléphonique publique;
- d’un protocole d’échange de paquets audio sur IP.

Qui utilise BitTorrent?

Qui utilise BitTorrent?

Logiciel de partage de fichiers composé

- d’un protocole d’échange de paquets de données sur IP.

Qui utilise BitTorrent et Skype?

Qui utilise BitTorrent et Skype?

A tout moment il est possible de connaître l’adresse IP

- d’un utilisateur de Skype;
- de machines impliquées dans un téléchargement BitTorrent.

On peut dire qui télécharge quoi/depuis où!

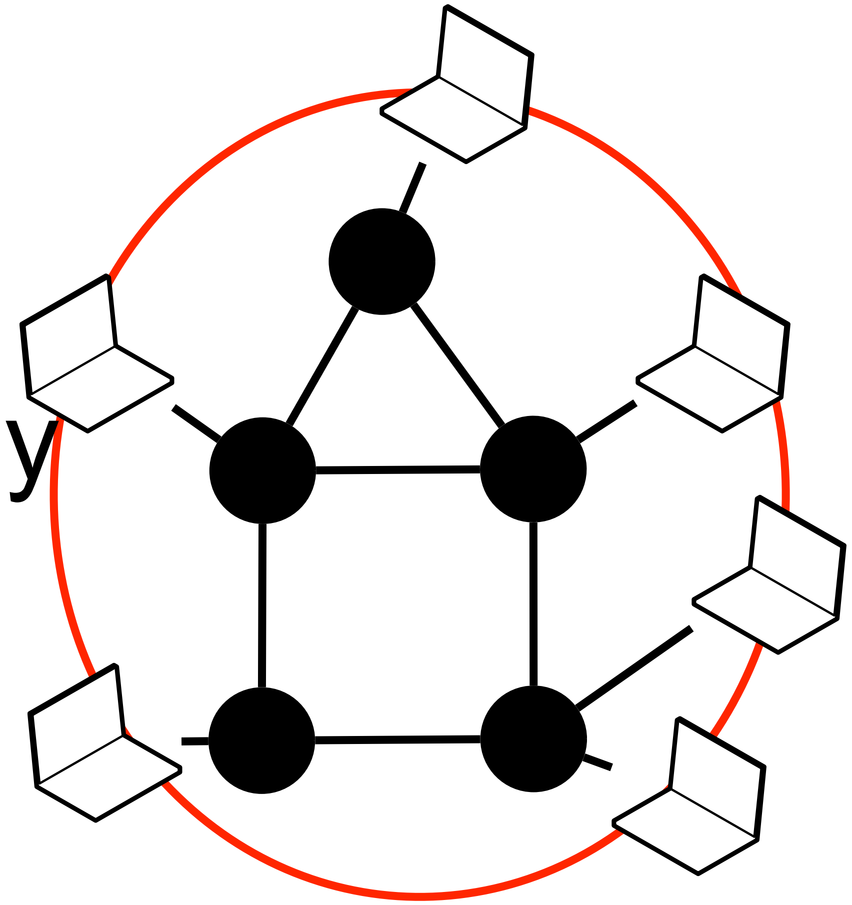
On peut dire qui télécharge quoi/depuis où!

Depuis chez soi

Overlay networking

Overlay network

- Constructed on top of another network, called the underlay
- Nodes in the overlay appear to be connected independently of the underlay



Definitions

- Peer
  - A node involved in forming the overlay (can be a computer, an end-user, an application…)
- Leecher
  - A peer that is both client and server
- Seed
  - A peer that is only server

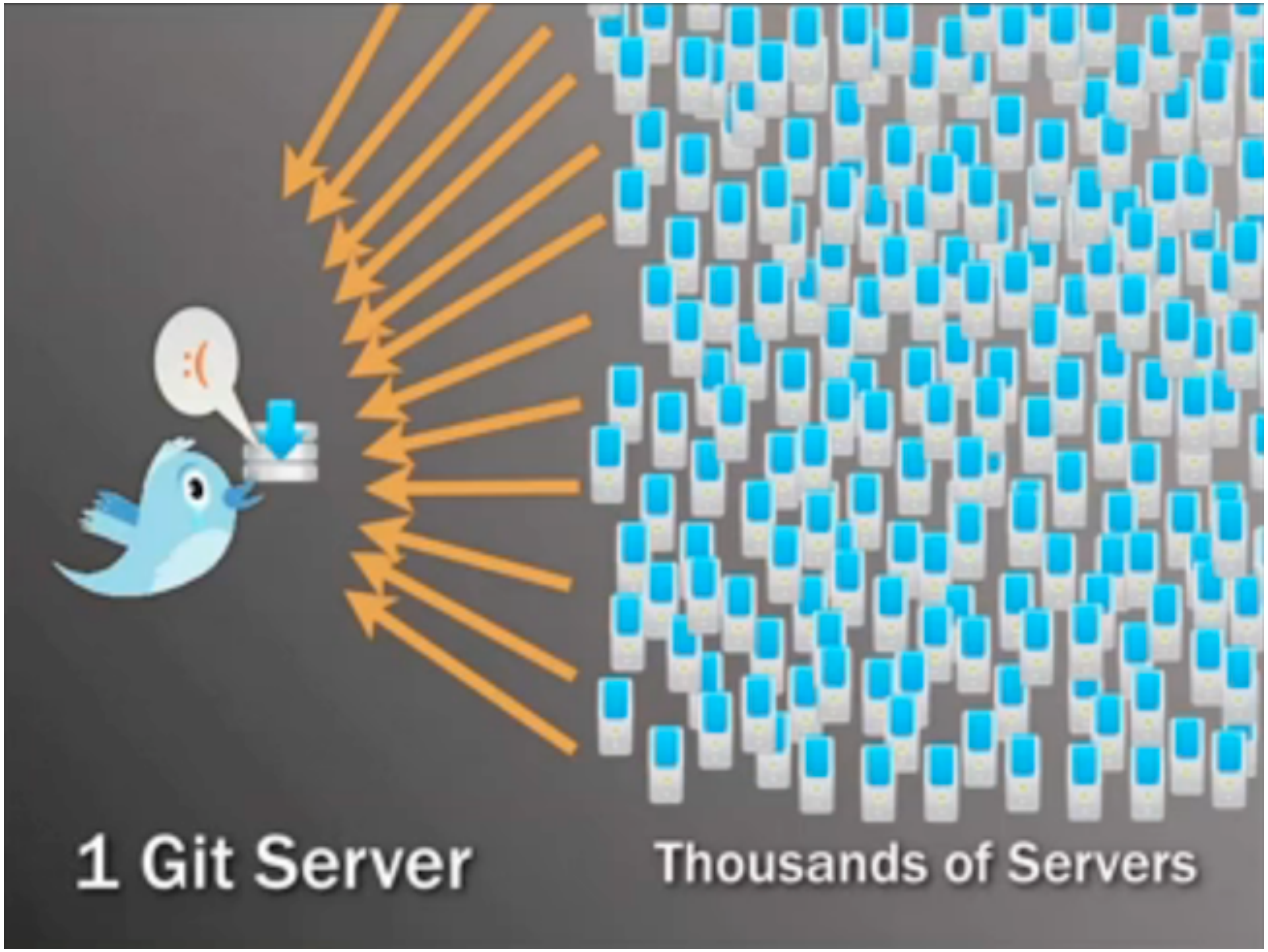
Definitions (contd.)

- Peer-to-peer (P2P) application
  - No general definition
  - Specific to an application
  - Every peer is client and server
  - Peers form an overlay network
- In general, we define P2P application as overlay network formed by end-users

P2P

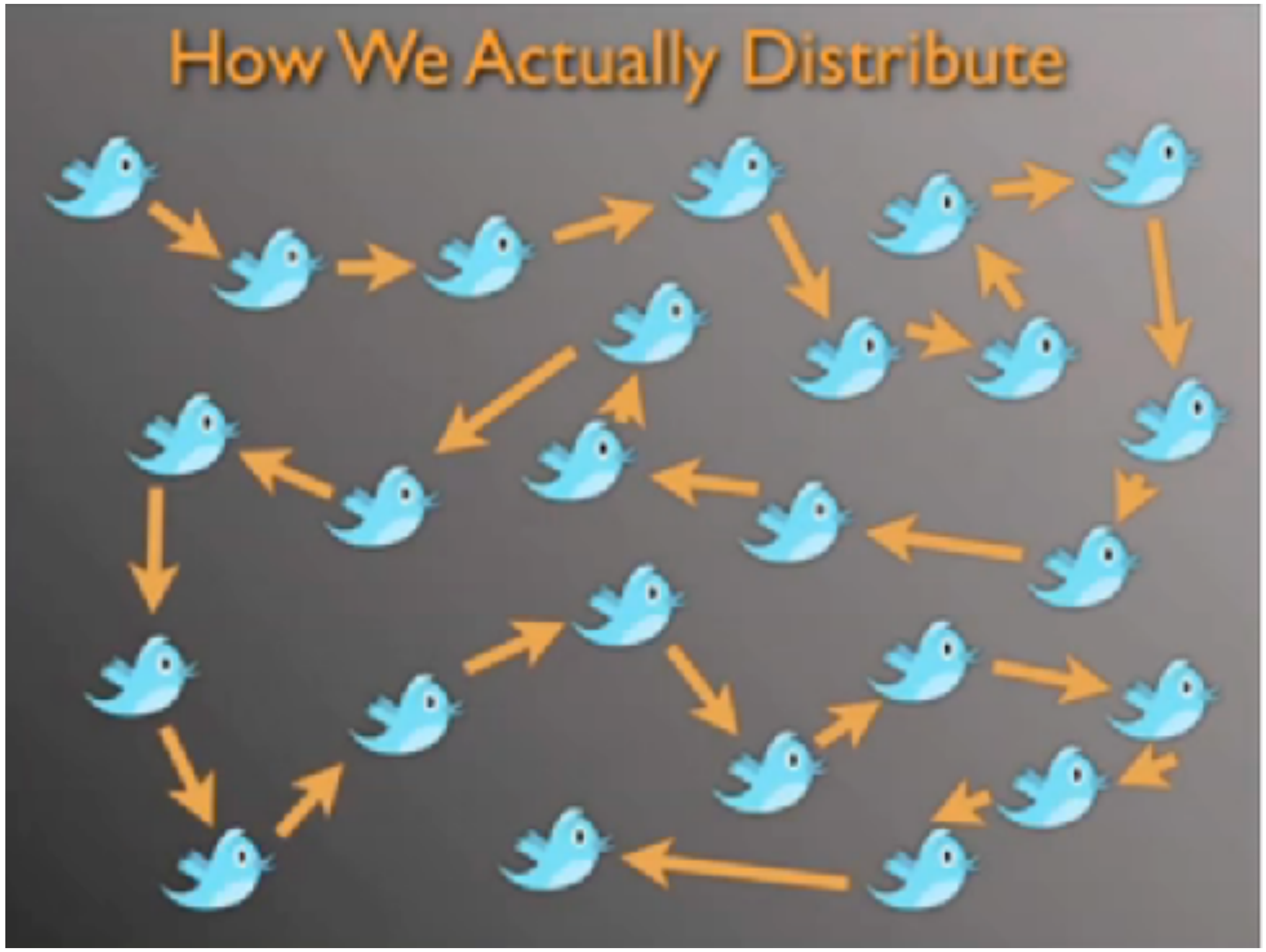
- P2P applications capitalize on any resource from anybody
  - CPU
  - Bandwidth
  - Storage

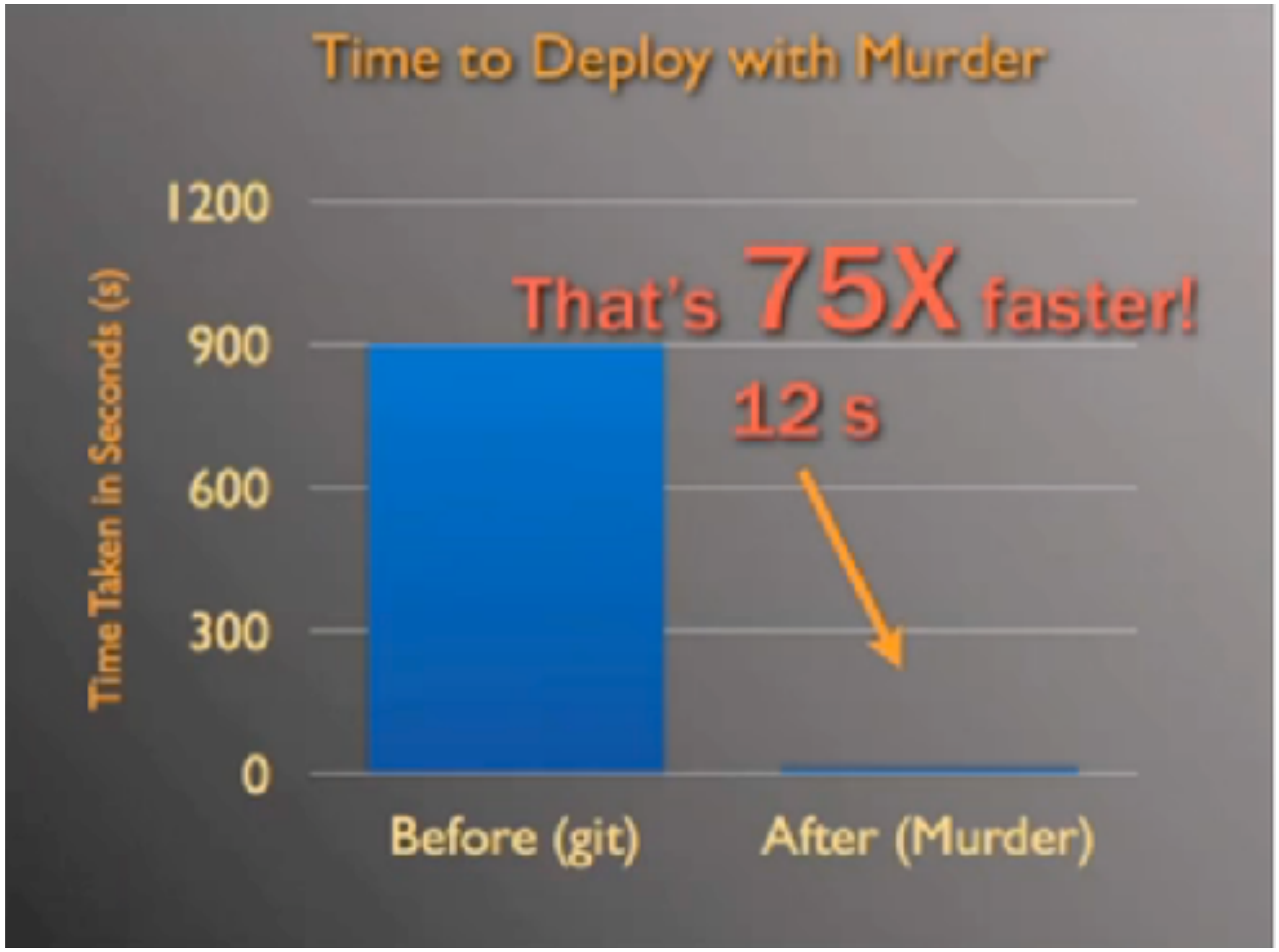
Before Murder



credit: https://blog.twitter.com/2010/murder-fast-datacenter-code-deploys-using-bittorrent

With Murder





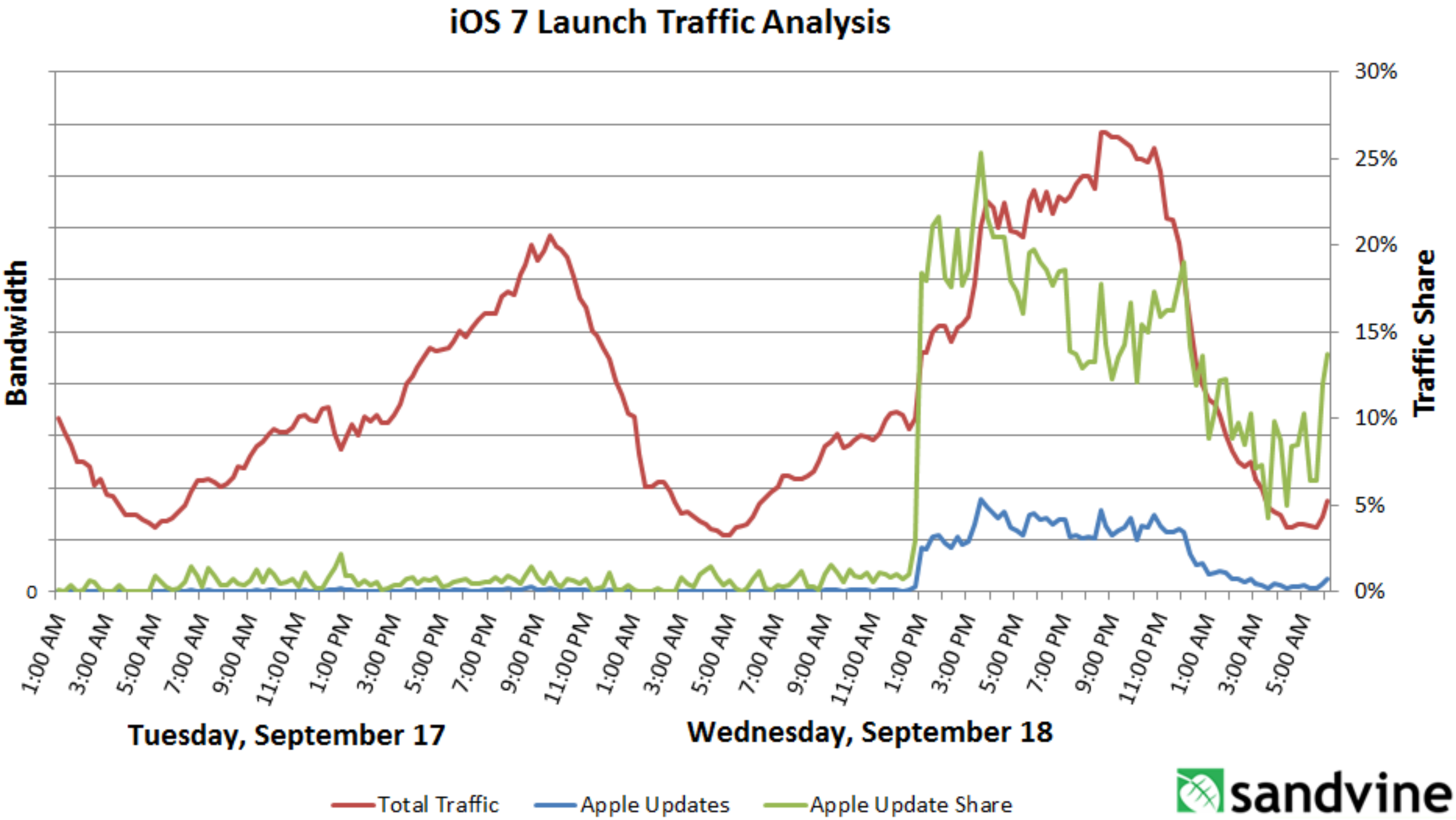
credit: https://blog.twitter.com/2010/murder-fast-datacenter-code-deploys-using-bittorrent

Content replication

Definitions

- Service capacity
  - Number of peers that can serve a content
    - = 1 in client-server, constant with time
- Flash crowd of n
  - Simultaneous request of n peers (e.g., soccer match, iOS update…)
- Piece/chunk/block
  - Element of a partition of the content
    - Each piece can be independently retrieved
    - The union of pieces forms the content

Definitions



http://www.internetphenomena.com/2013/09/traffic-spotlight-ios-7-launch/

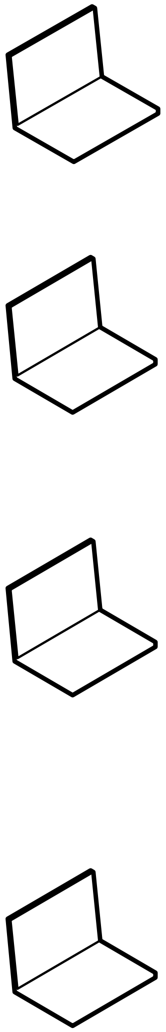
Interest of P2P to replicate contents

- Service capacity grows up exponentially with time
  - Average download time for a flash crowd *n* is then in *log(n)*
  - Average download time decreases in $\frac{1}{\#\text{ of pieces}}$ when the number of pieces increases
    - if we ignore the overhead

Content transfer model

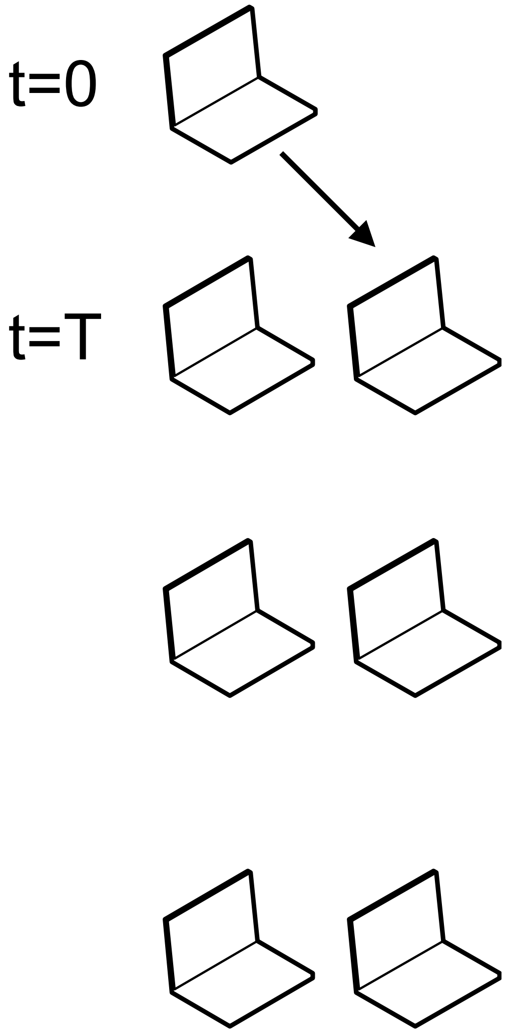
- Simple deterministic model
  - Each peer serves only one peer at a time
  - The unit of transfer is the content
  - n-1 peers want the content, with $n=2^k$
- T is the time to complete an upload
  - T=s/b, s content size, b upload capacity
- Peer selection strategy with Binary tree
  - global knowledge

Capacity C of the service

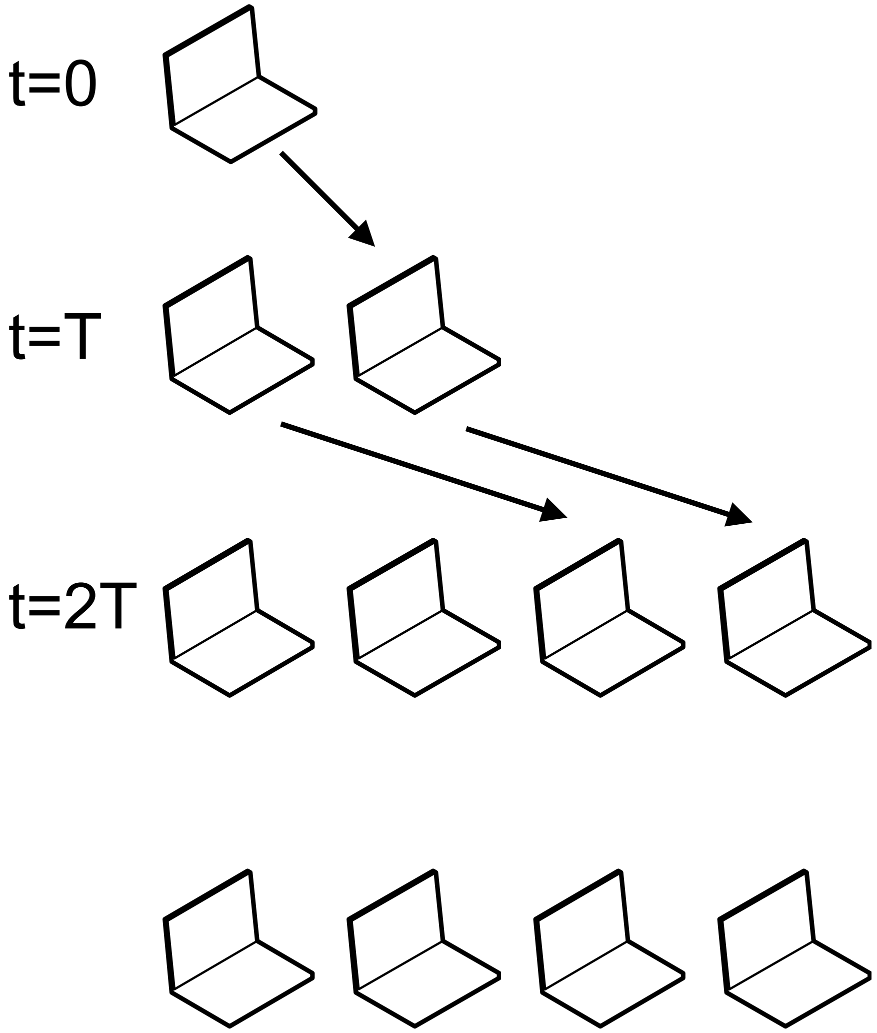


t=0

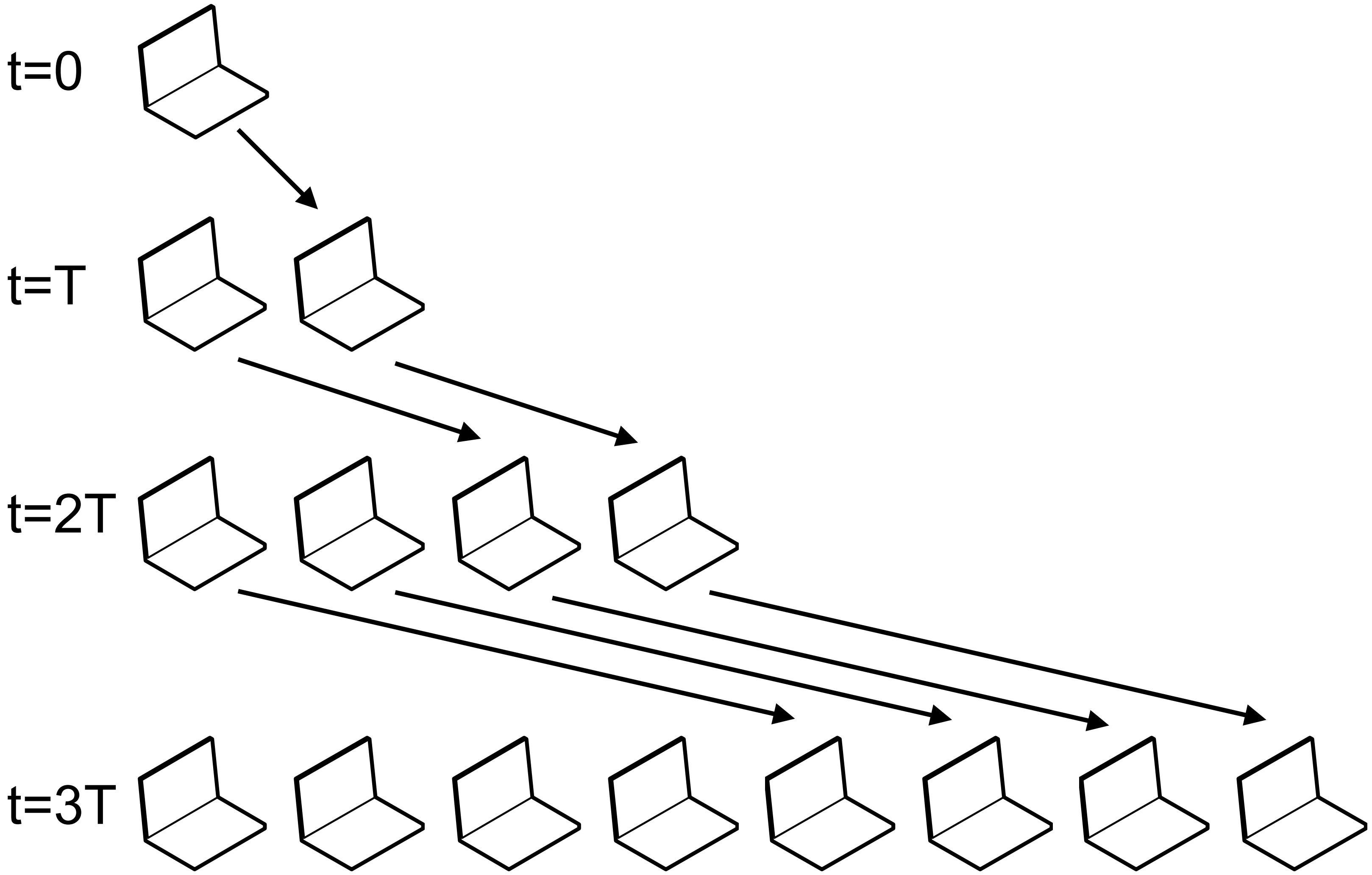
Capacity C of the service



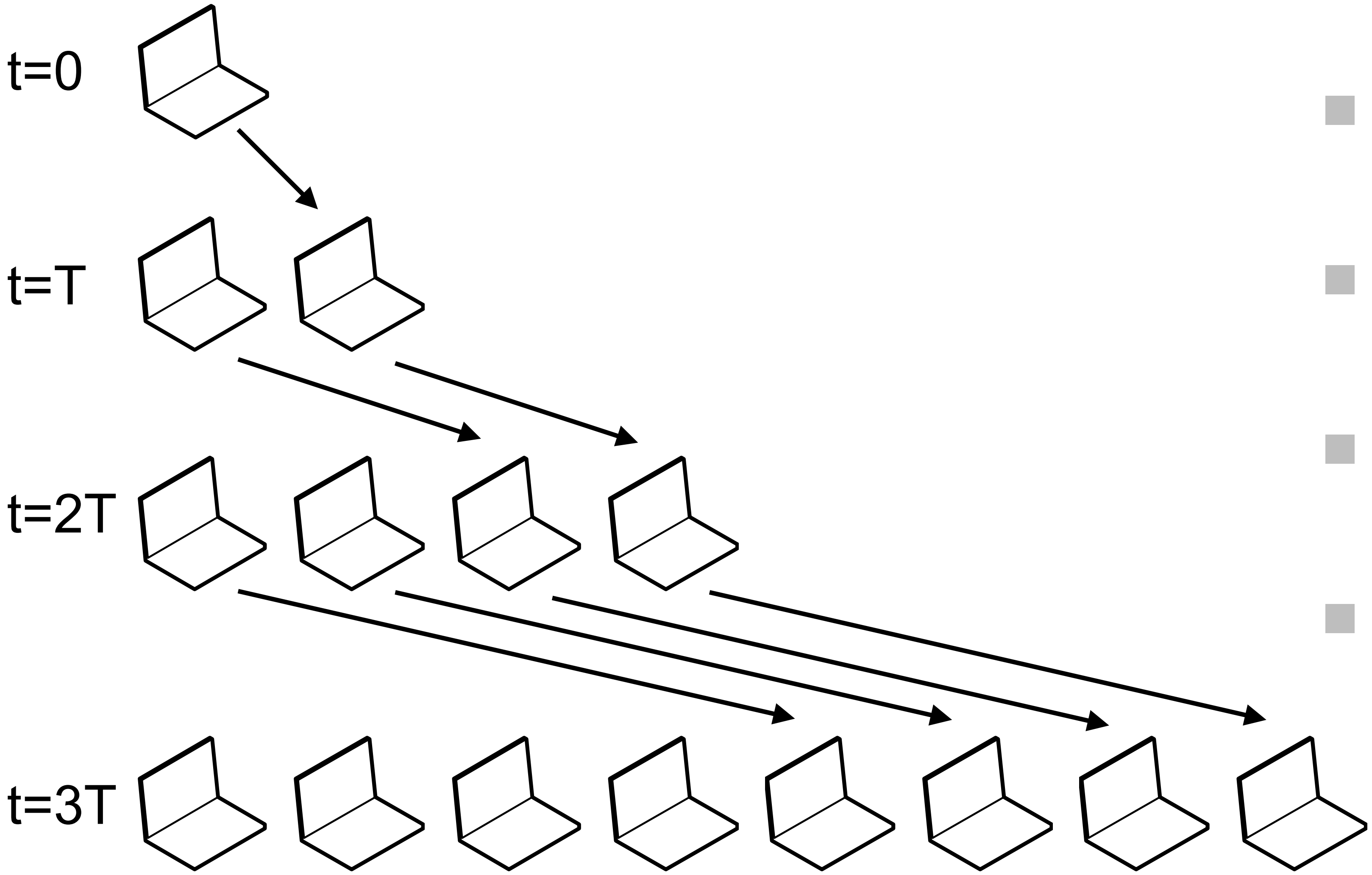
Capacity C of the service



Capacity C of the service



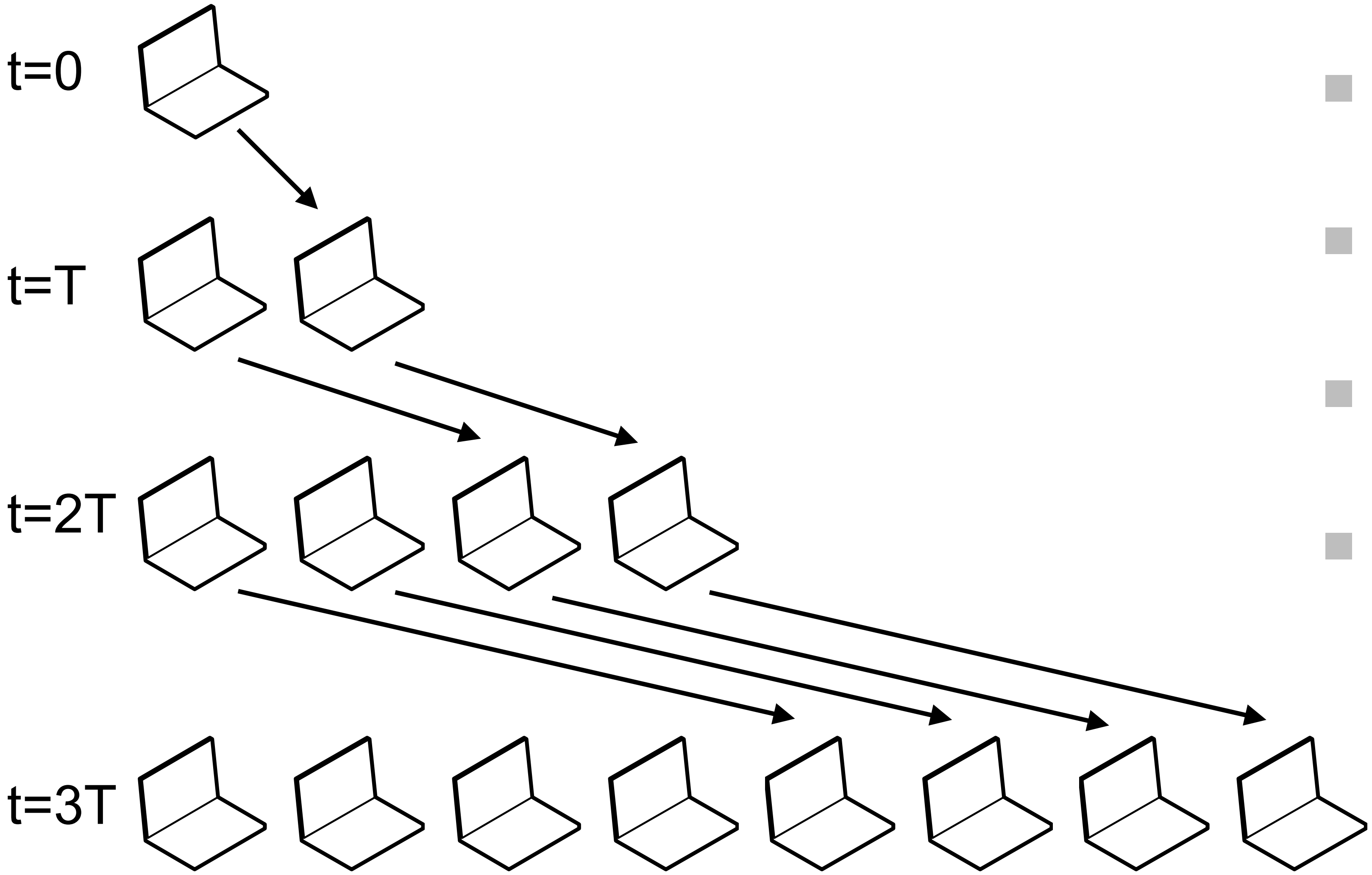
Capacity C of the service



- t=0 => C = 2^0 peers
- t=T => C = 2^1 peers
- t=2*T => C = 2^2 peers
- …
- t=i*T => C = 2^i peers

➡ C = $2^{t/T}$ peers

Finish time



- seed only at time t = 0
- 2^0 peers finish at t = T
- 2^1 peers finish at t=2T
- …
- 2^{k-1} peers finish at t=k*T

➡ content transferred to all peers at t = k*T = T * $\log_2(n)$ vs n*T in client-server

Can we speed up transfers?

Piece transfer model

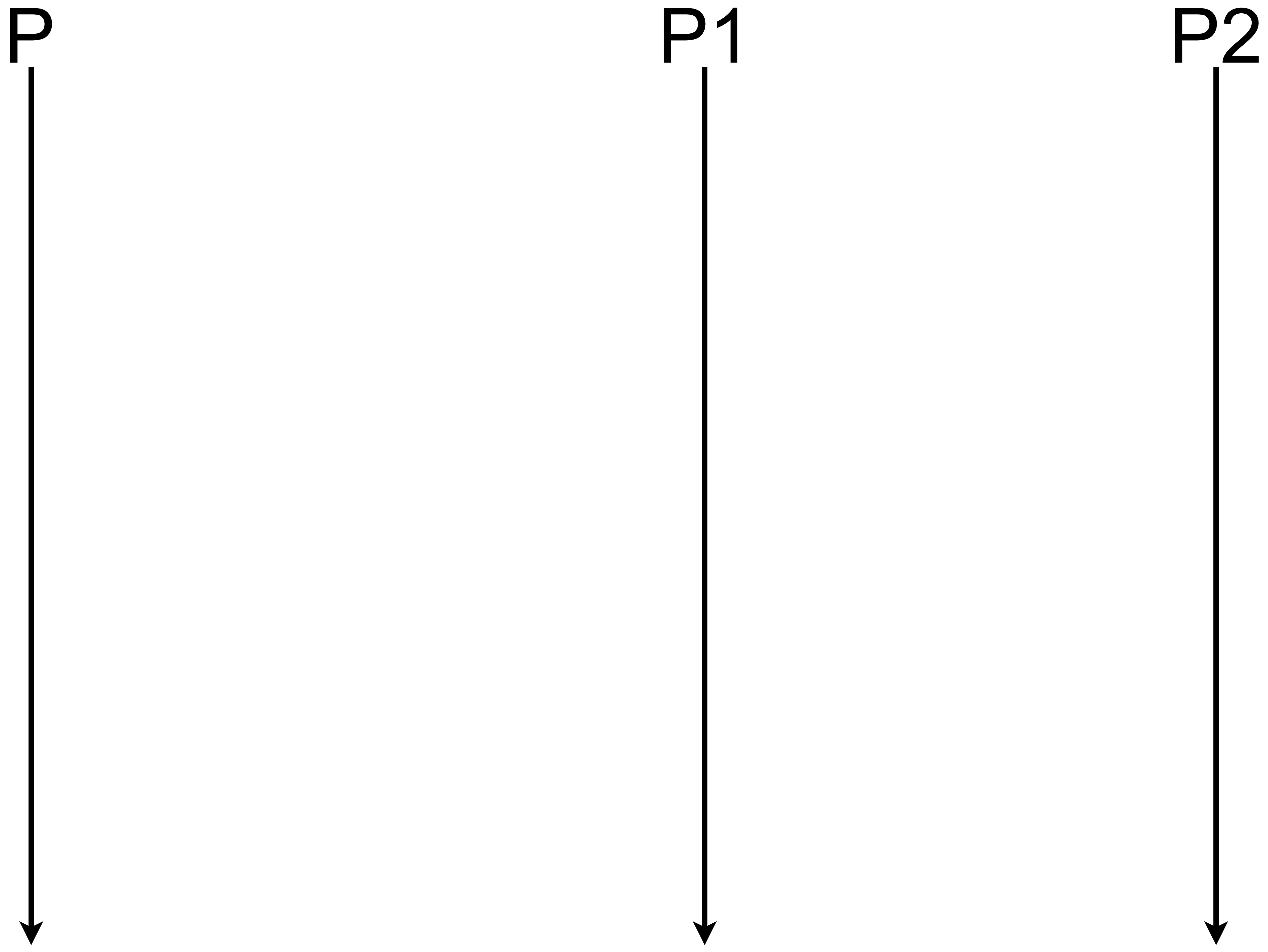
- Same as before but the transfer unit is the piece instead of the content
  - a content is divided into m equal size pieces
  - m > k
  - Piece downloaded in T/m

➡content transferred to all peers at $t = T * 1/m * \log_2(n) + T$ vs $T * \log_2(n)$ without piece transfer vs $n*T$ in client-server

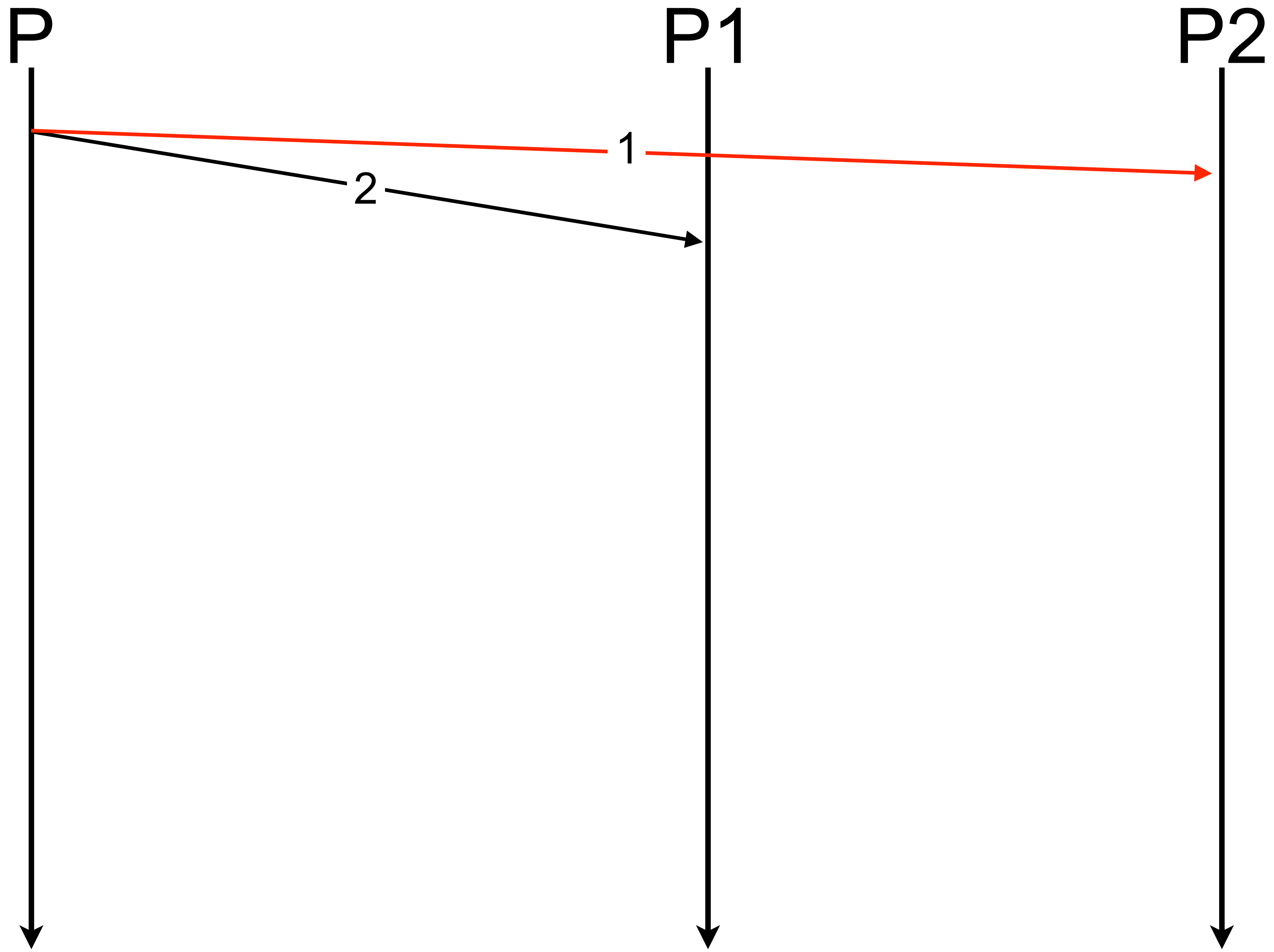
Parallel downloads

- Download from several peers in parallel
- Strategy
  - request one piece from every server with the content
  - request another piece from the server as soon as the requested piece has been obtained
    - performance is optimal when servers are always busy delivering a piece of data

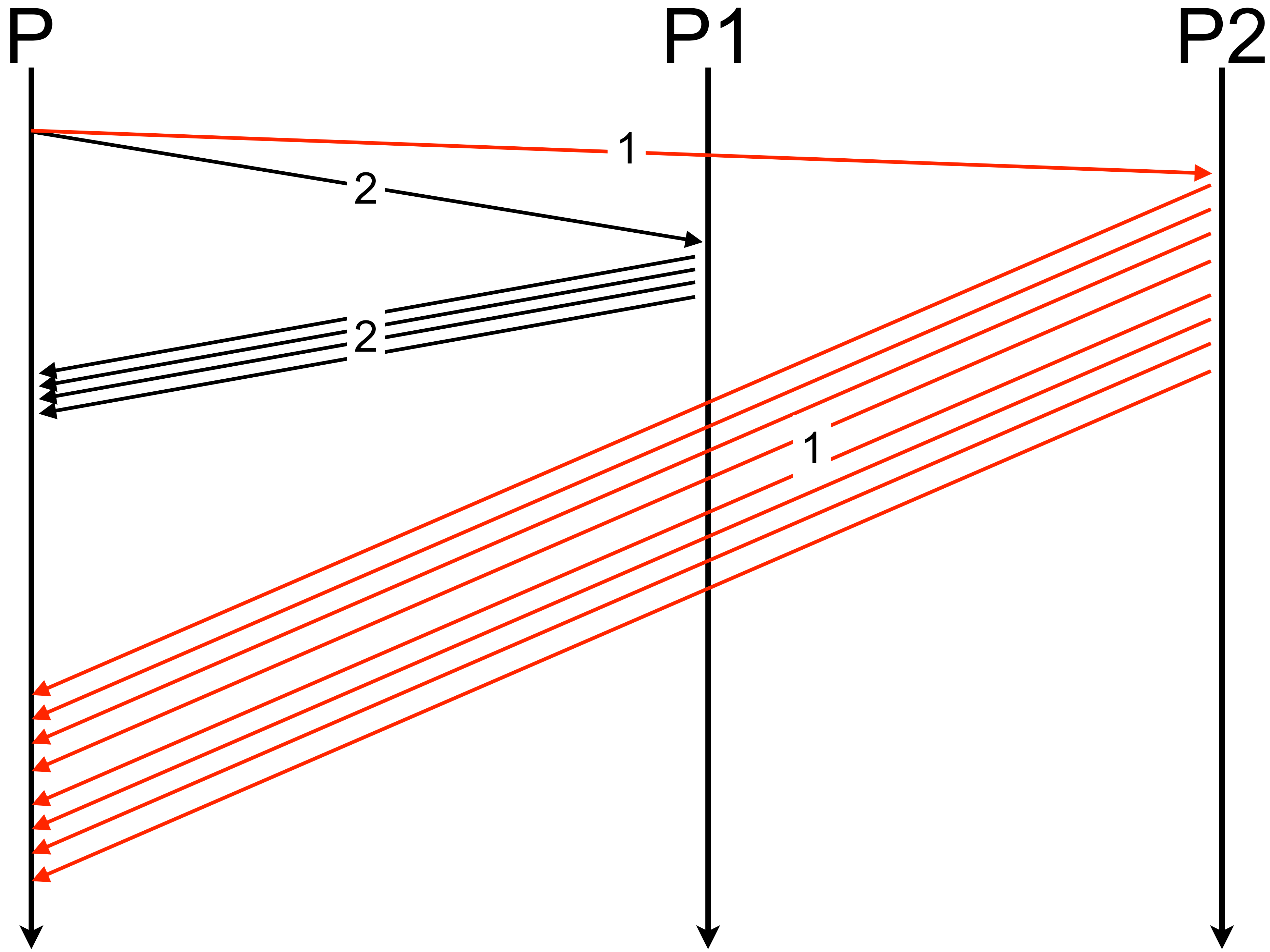
Parallel downloads (contd.)



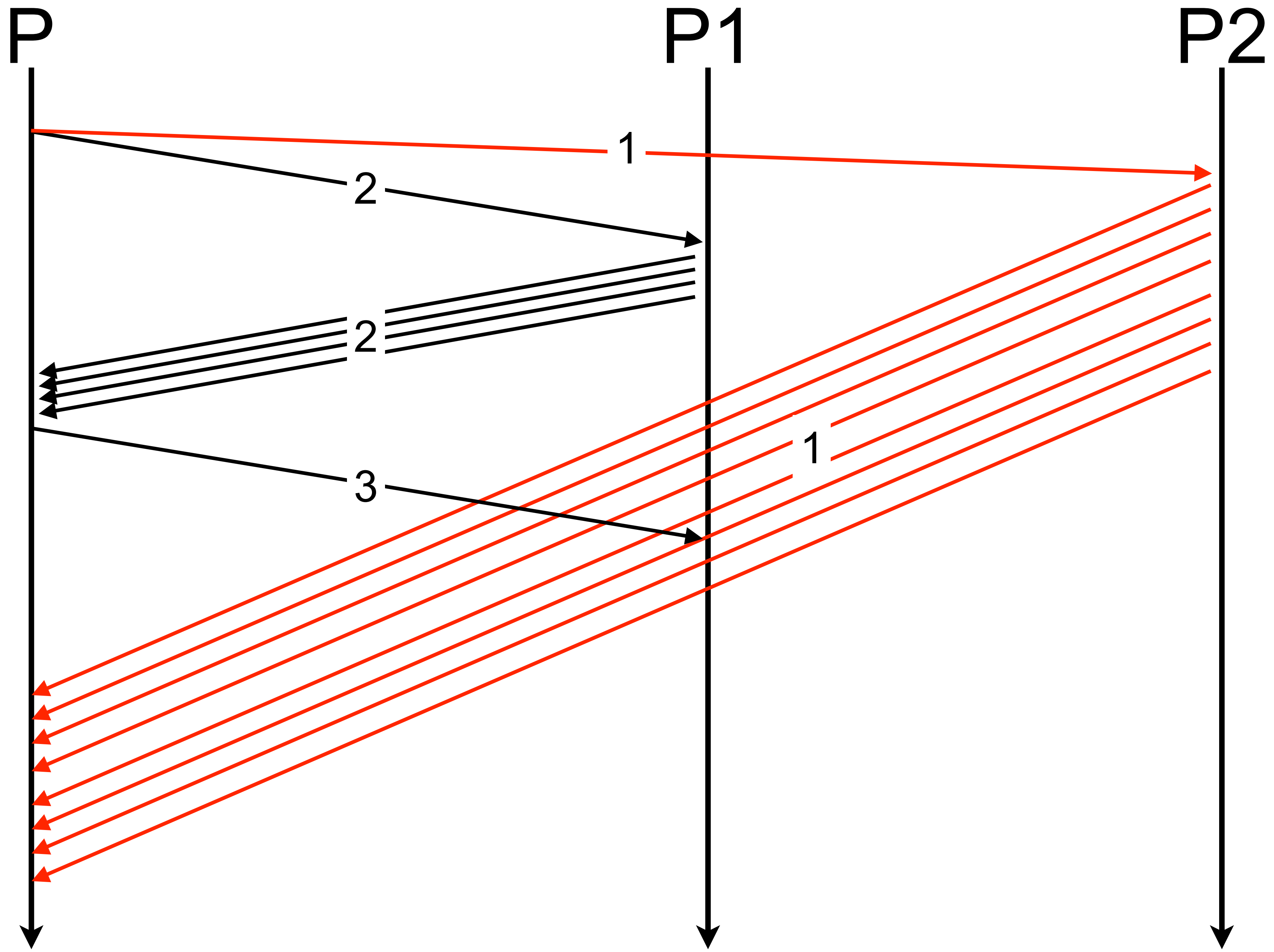
Parallel downloads (contd.)



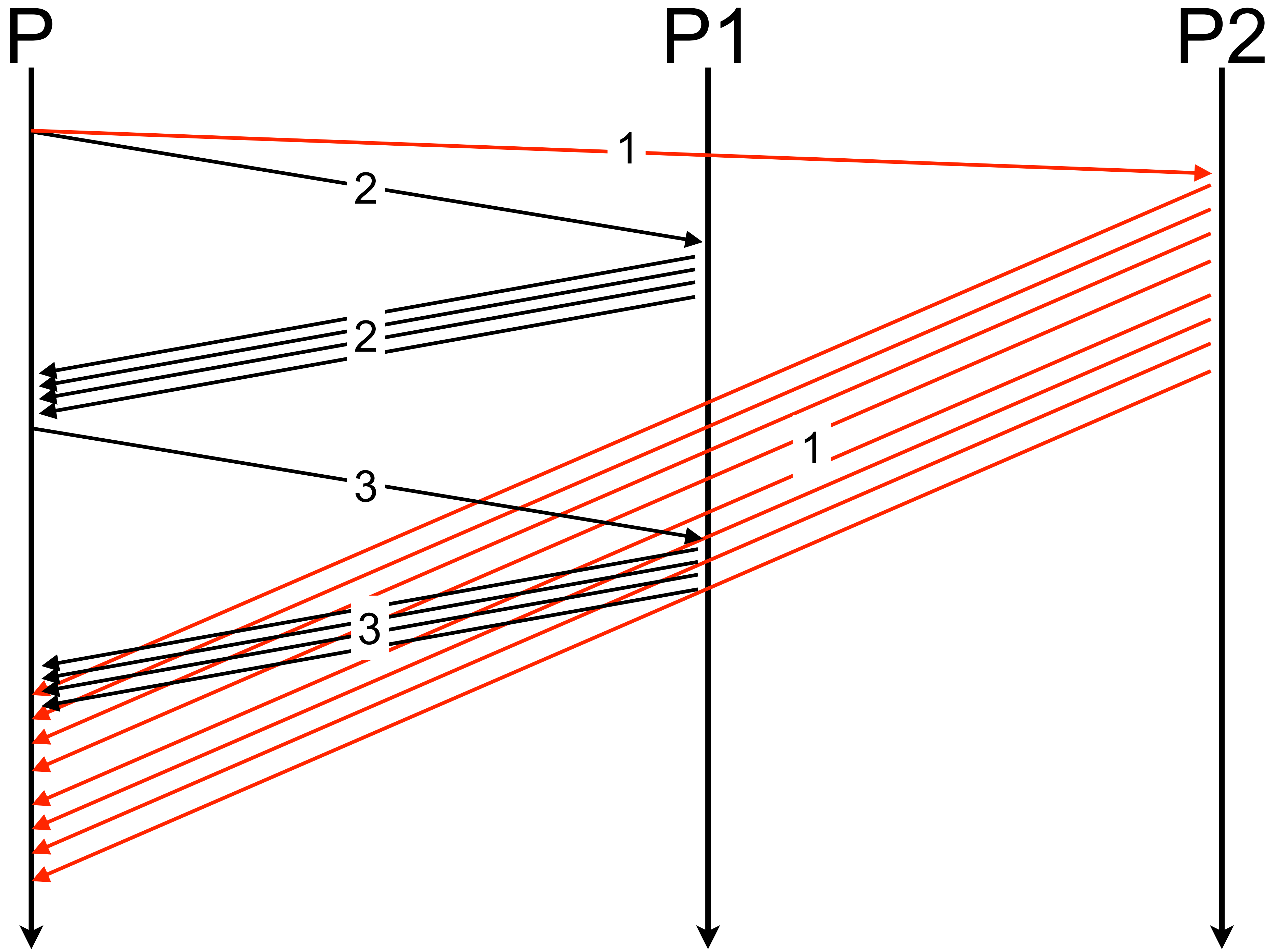
Parallel downloads (contd.)



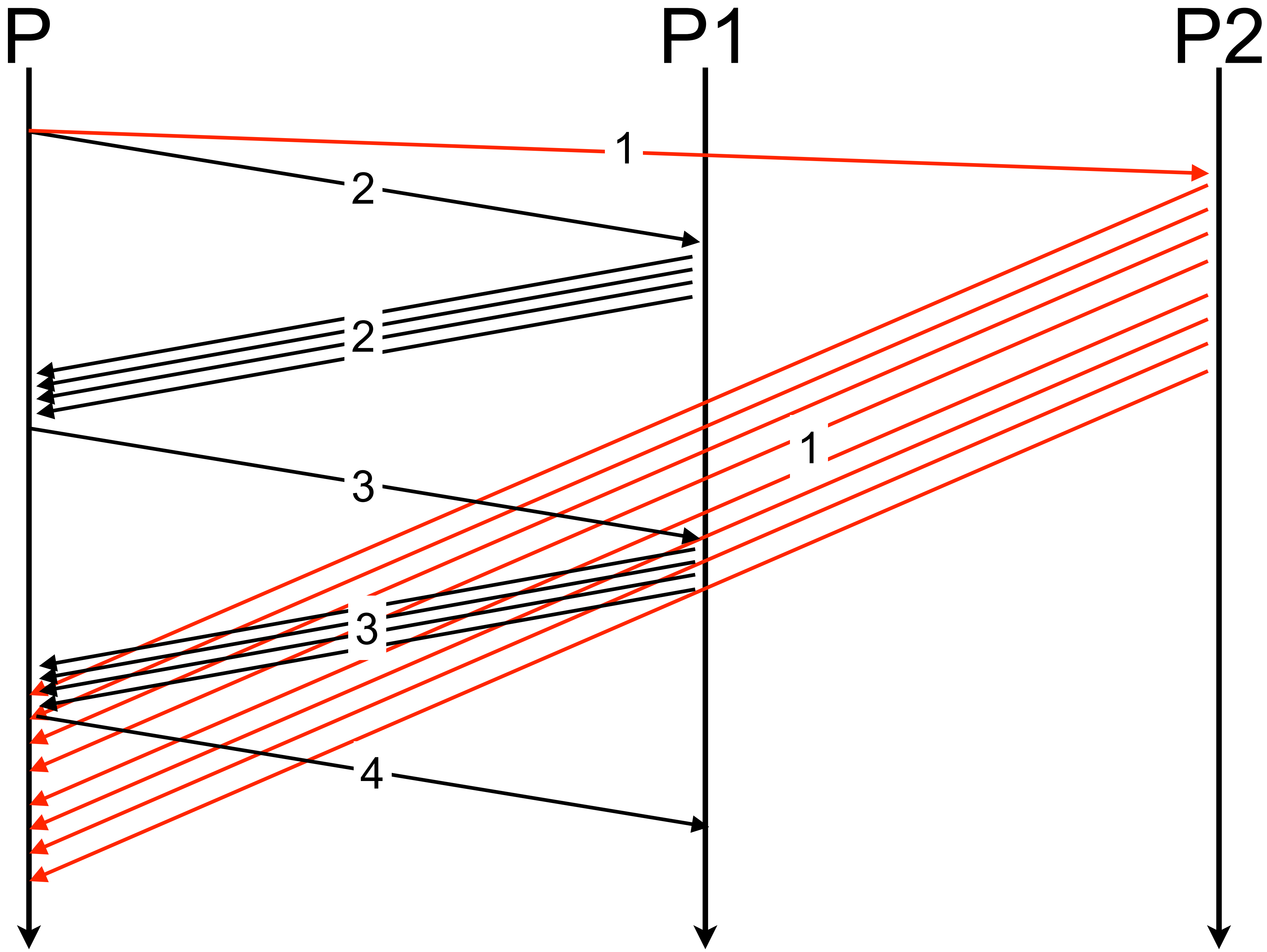
Parallel downloads (contd.)



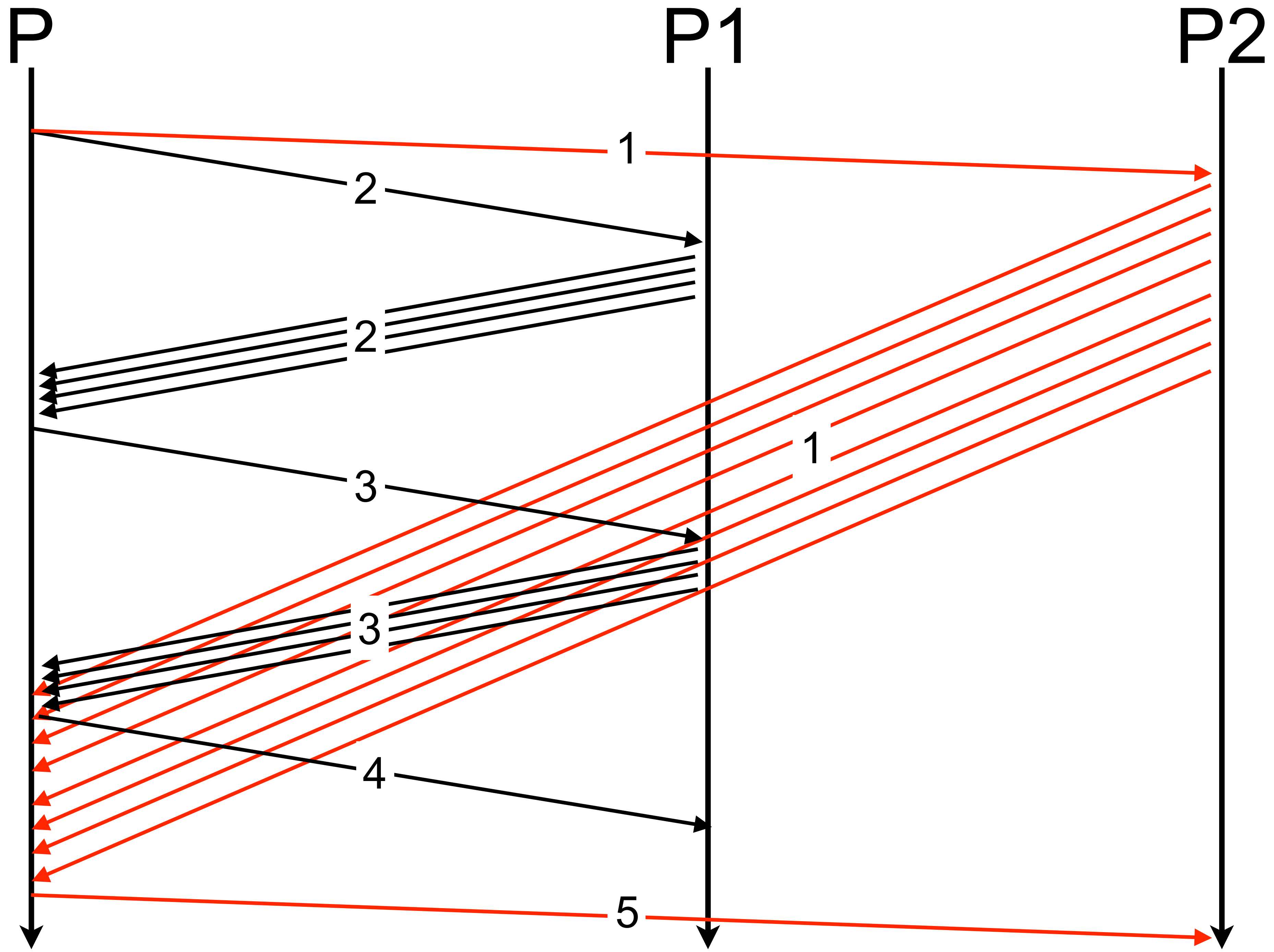
Parallel downloads (contd.)



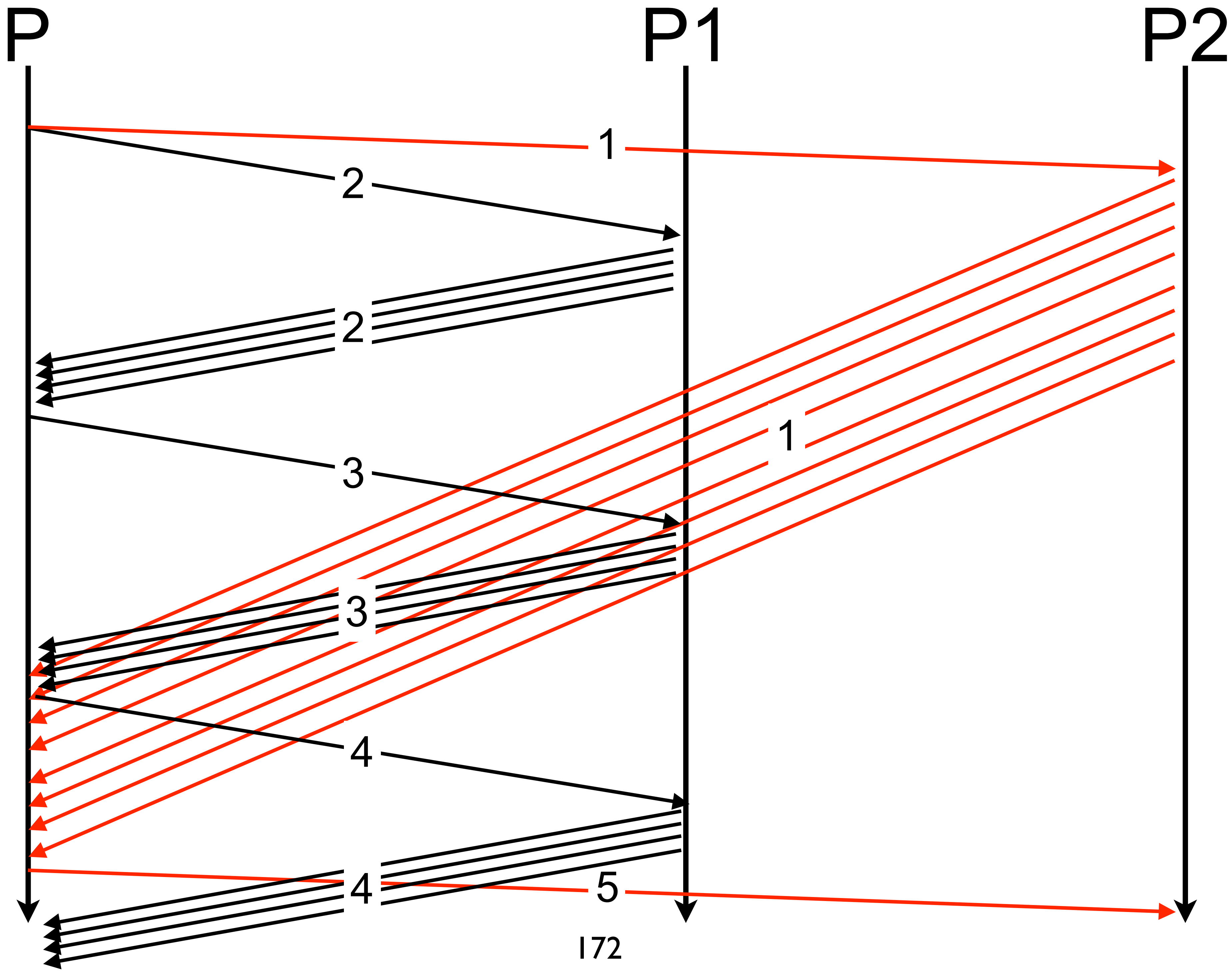
Parallel downloads (contd.)



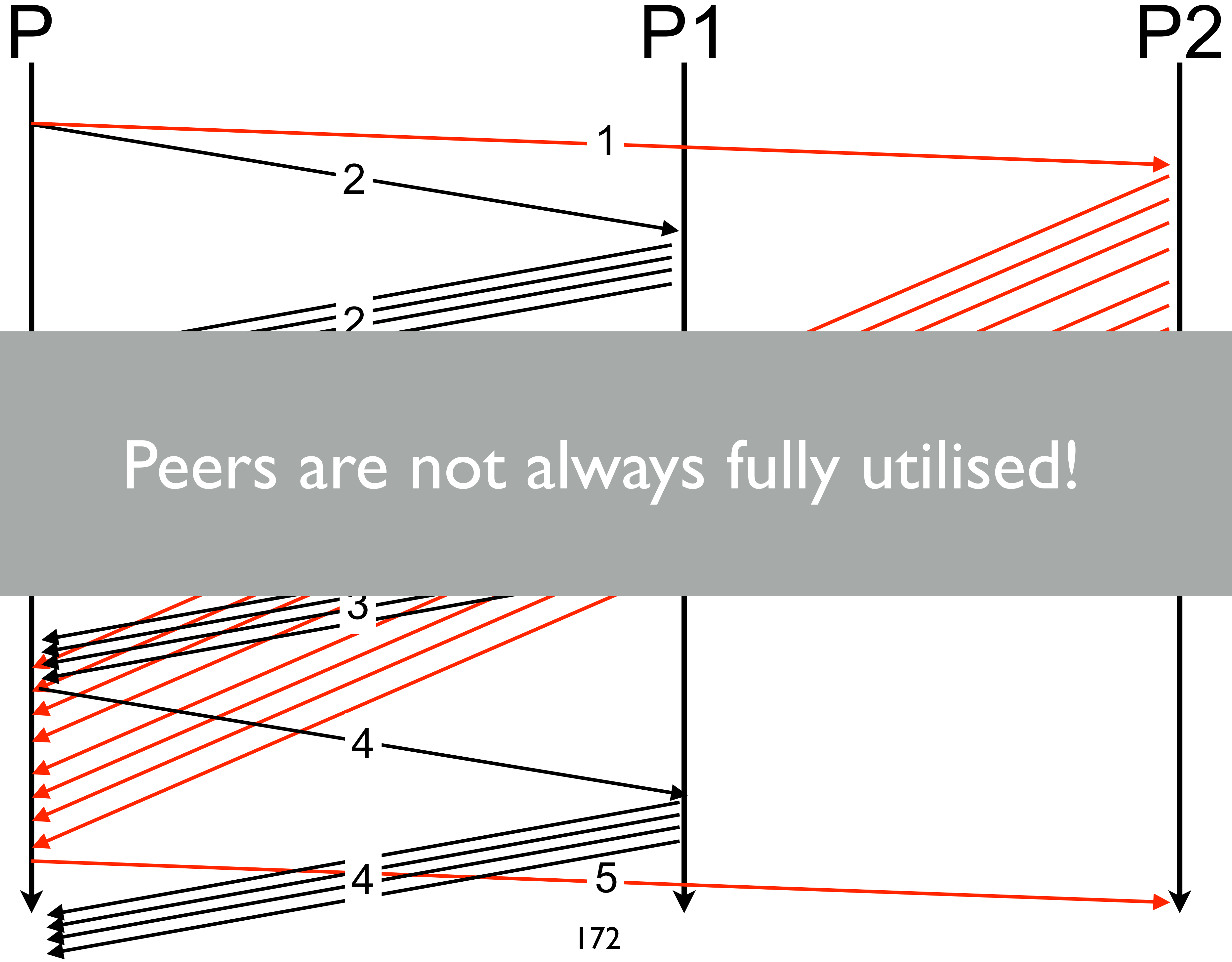
Parallel downloads (contd.)



Parallel downloads (contd.)

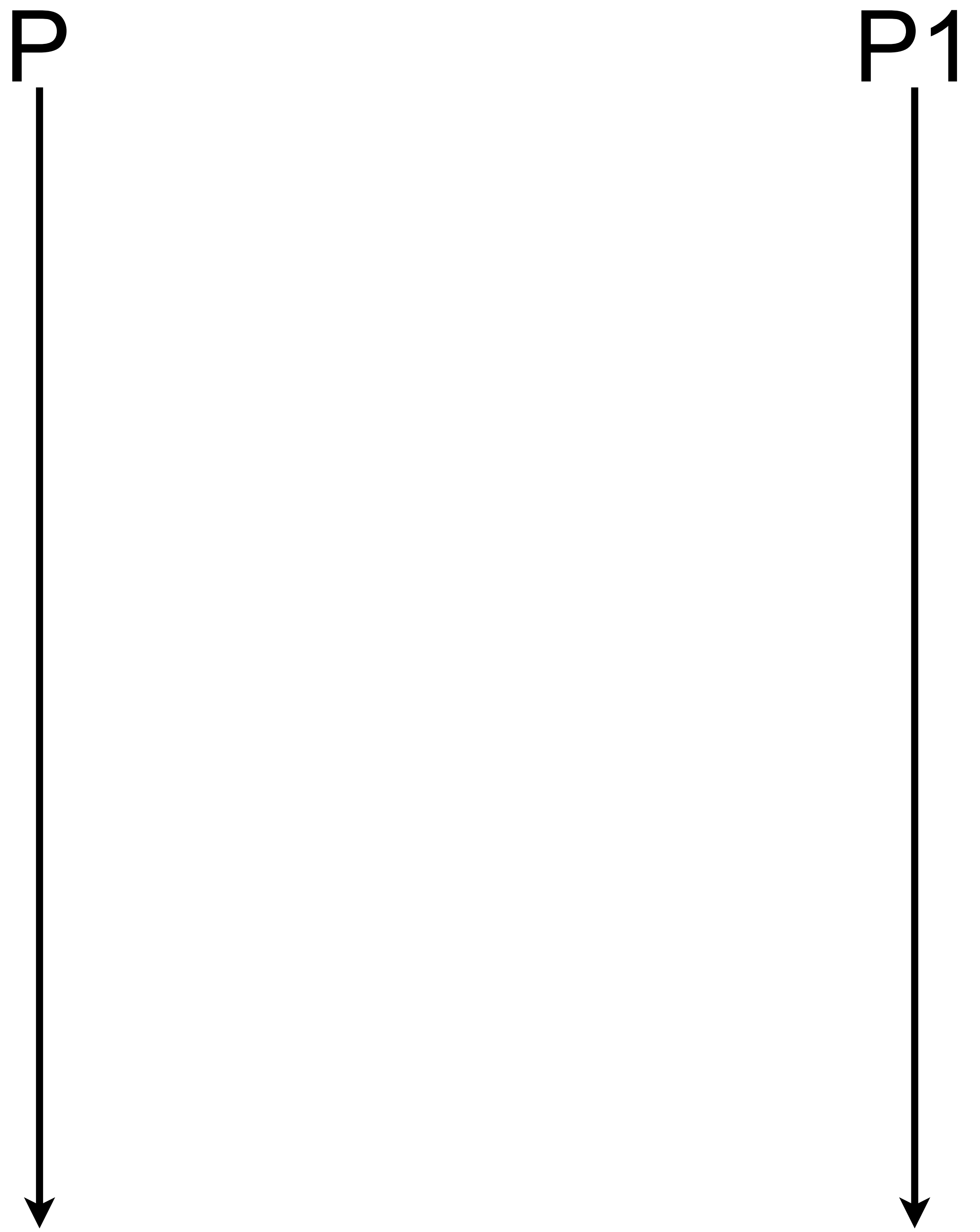


Parallel downloads (contd.)



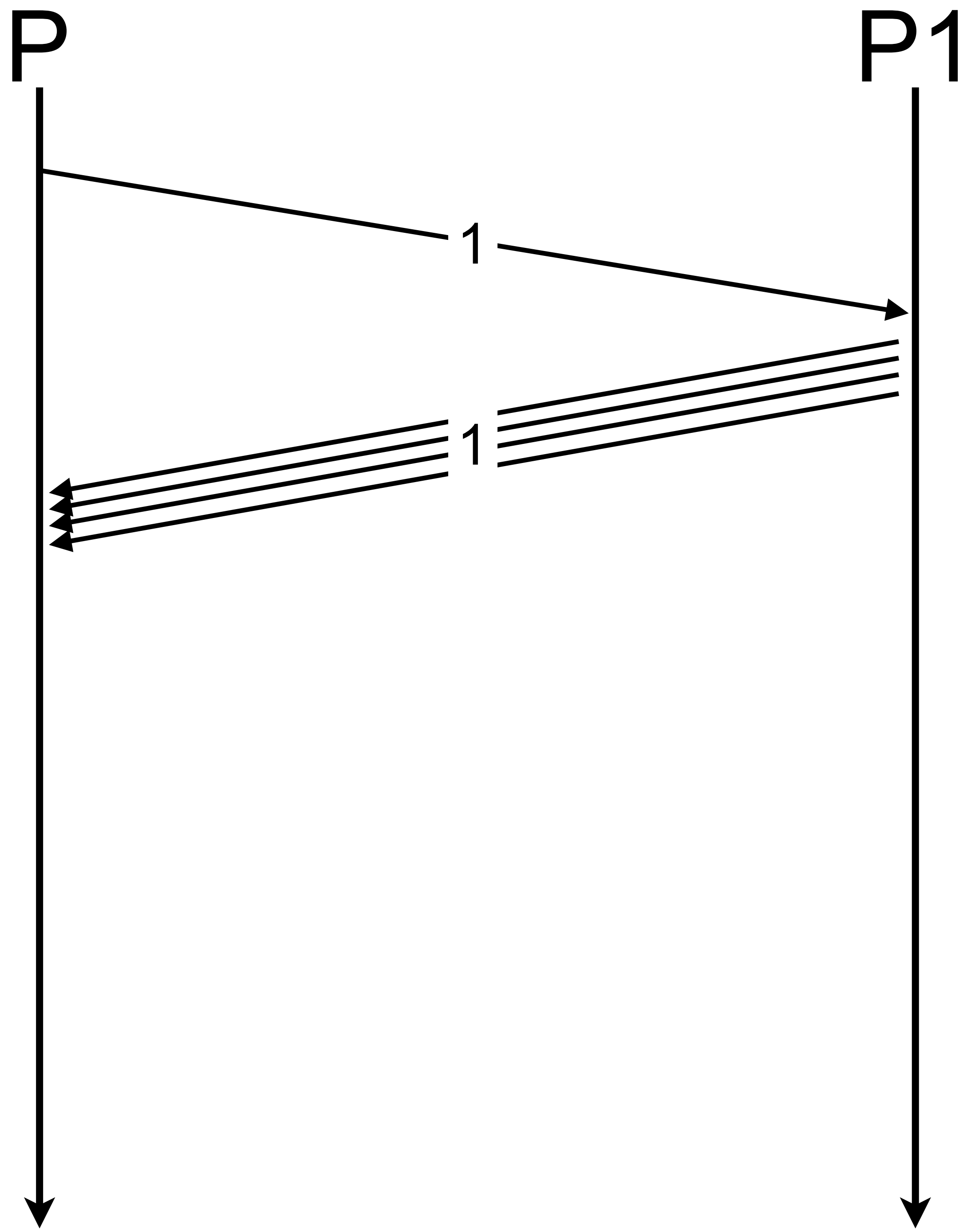
Peers are not always fully utilised!

Pipelining



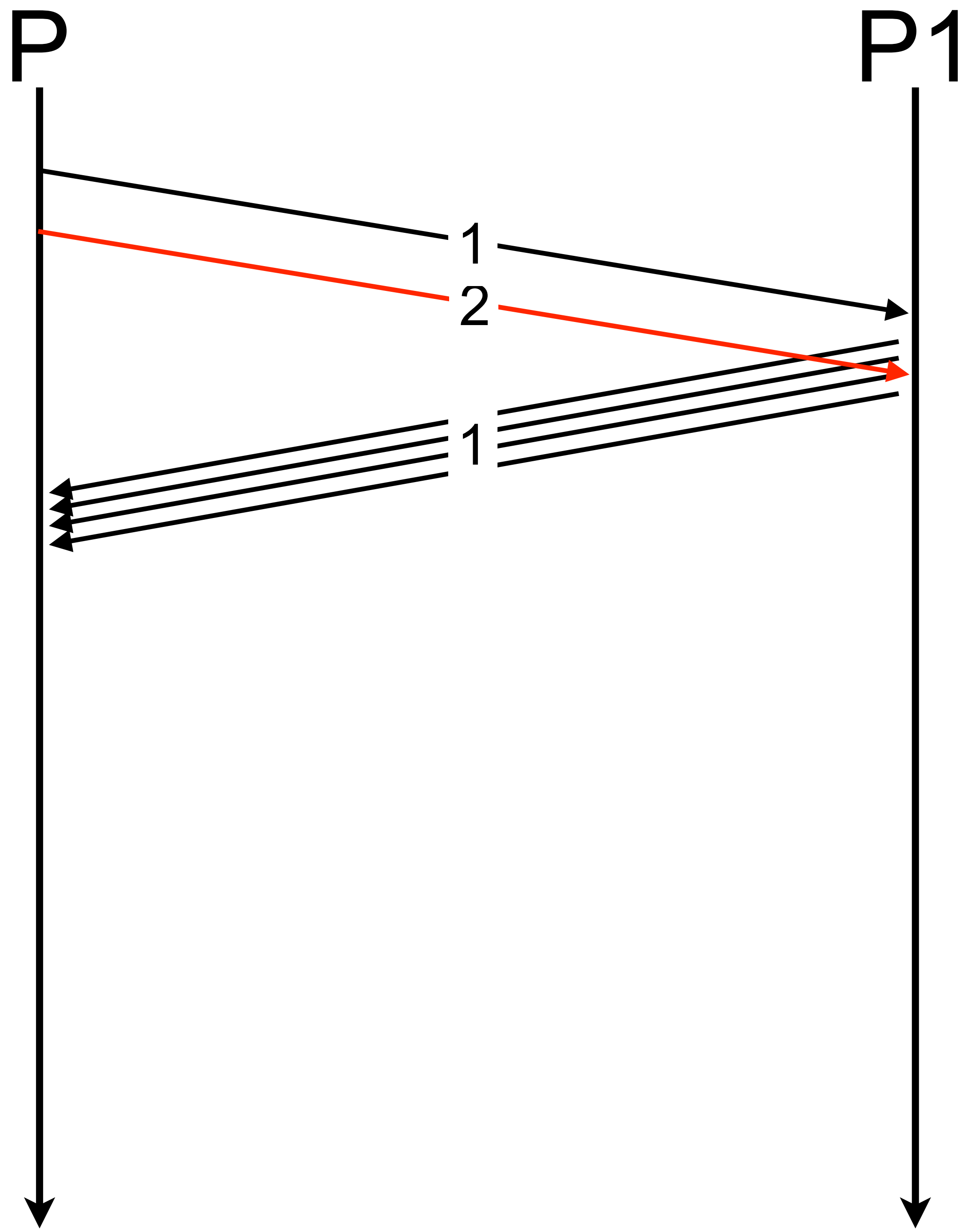
- Keep enough requests pending
- Send a new request before the end of the transmission of the piece being downloaded
  - need to roughly estimate the RTT

Pipelining



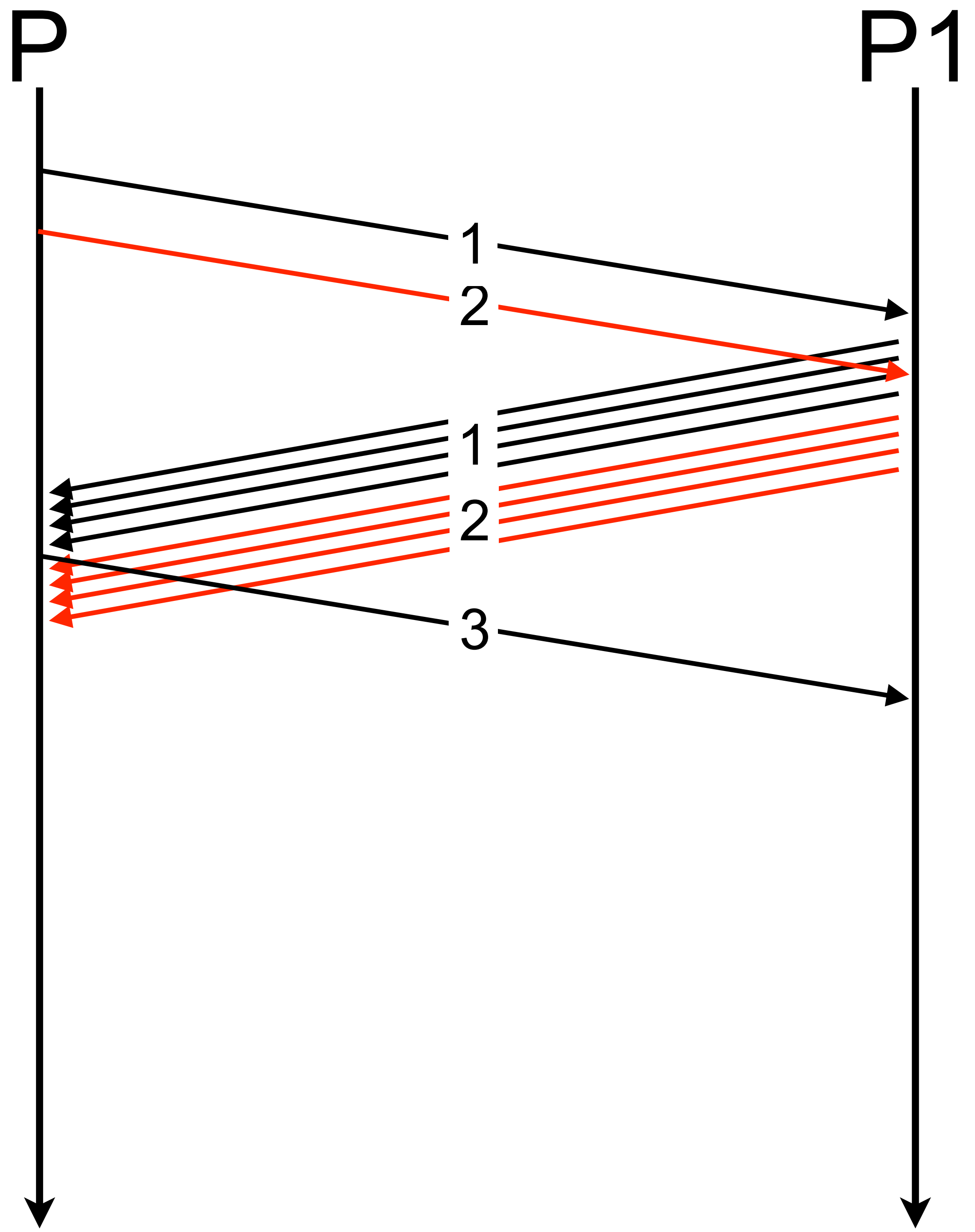
- Keep enough requests pending
- Send a new request before the end of the transmission of the piece being downloaded
    - need to roughly estimate the RTT

Pipelining



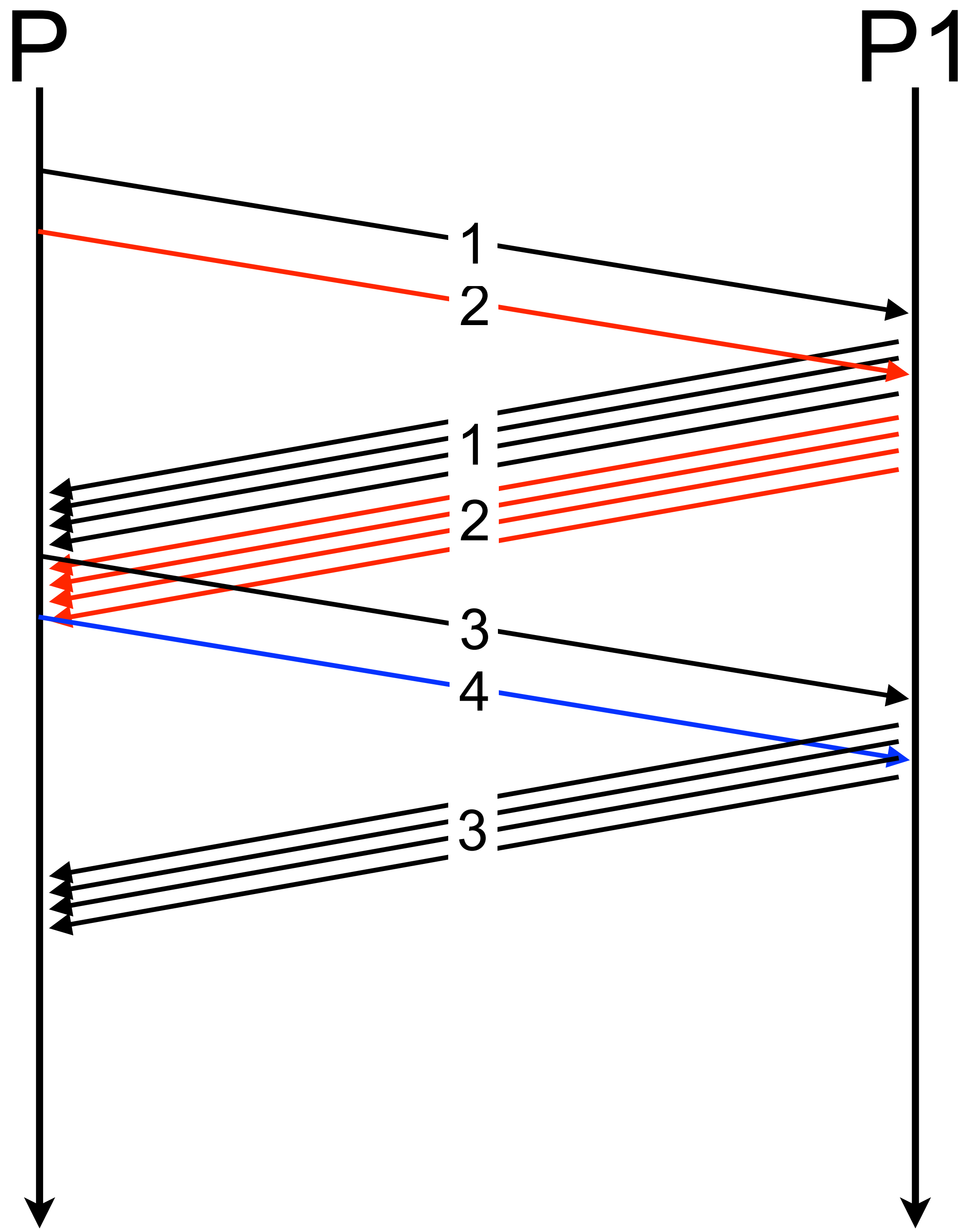
- Keep enough requests pending
- Send a new request before the end of the transmission of the piece being downloaded
  - need to roughly estimate the RTT

Pipelining



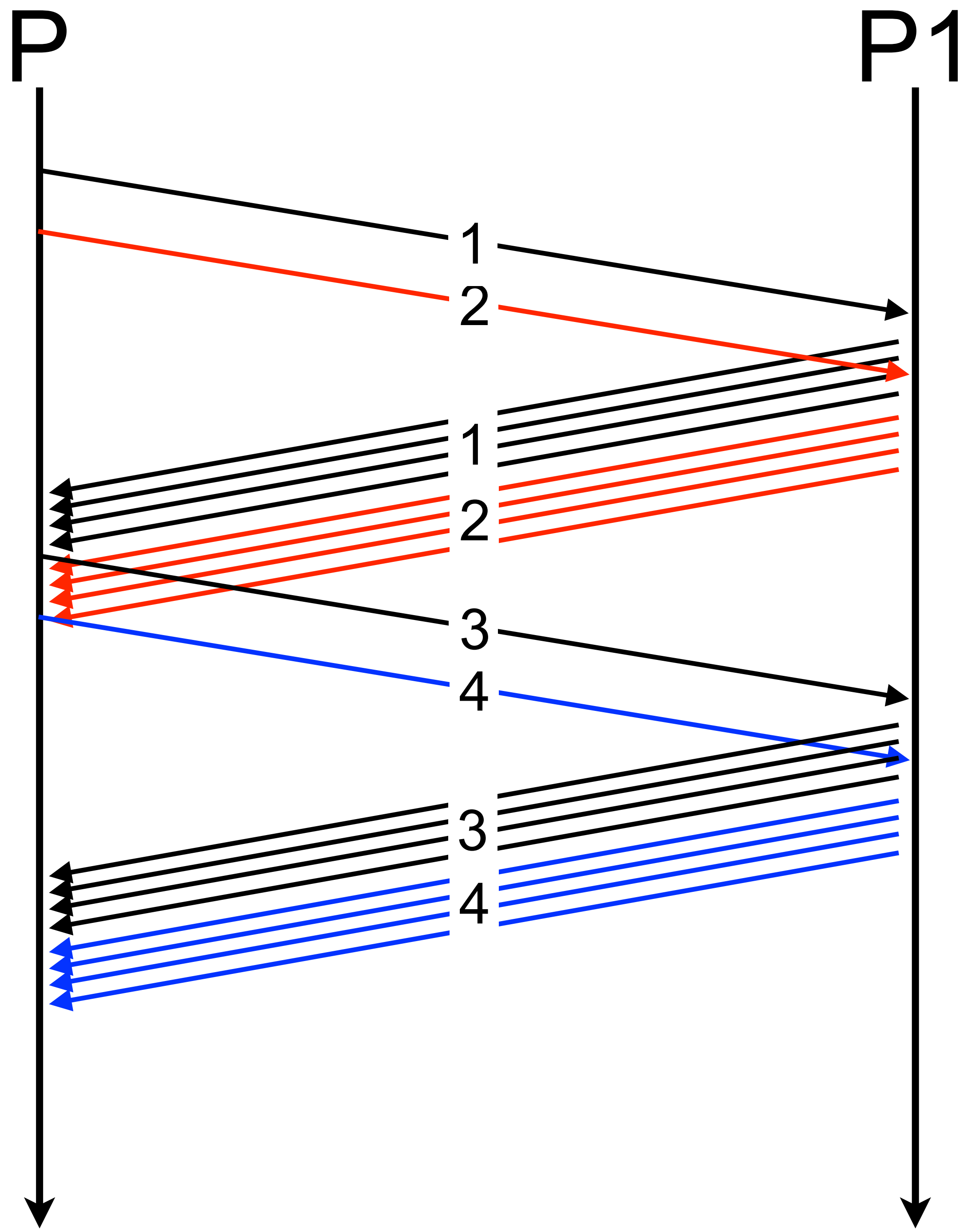
- Keep enough requests pending
- Send a new request before the end of the transmission of the piece being downloaded
  - need to roughly estimate the RTT

Pipelining



- Keep enough requests pending
- Send a new request before the end of the transmission of the piece being downloaded
  - need to roughly estimate the RTT

Pipelining



- Keep enough requests pending
- Send a new request before the end of the transmission of the piece being downloaded
  - need to roughly estimate the RTT