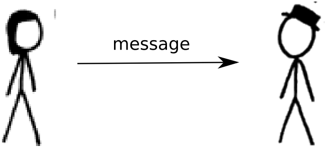
ΥΣ13 - Computer Security

Anonymous Communication

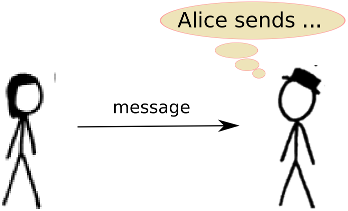
Κώστας Χατζηκοκολάκης

Anonymous communication

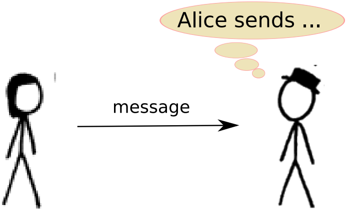
Anonymous communication



Anonymous communication

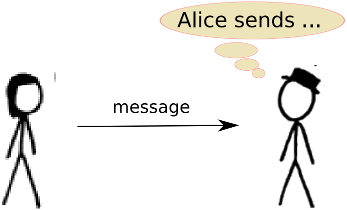


Anonymous communication



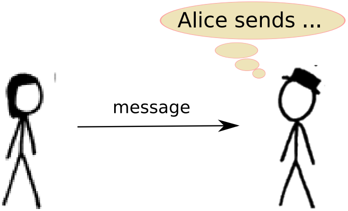
- Alice does not want Bob to know that she is the sender.

Anonymous communication



- Alice does not want Bob to know that she is the sender.
- Other properties?

Anonymous communication



- Alice does not want Bob to know that she is the sender.
- Other properties?
- Adversary model?

Why?

- Accessing censitive content
- Censorship resistance (eg. Great Firewall of China)
- Electronic voting
- Whistleblowing
- File sharing
- Profiling resistance
- Auctions / stock market

Why anonymity is difficult?

- `wolframalpha.com` : “who am I”?

Why anonymity is difficult?

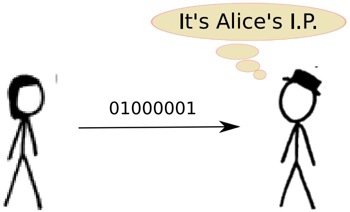
- `wolframalpha.com`: “who am I”?
- Sender’s IP address included in all IP packets
- Already enough to trace someone to ISP/region level
- Can be traced down to individuals using ISP’s logs
  (obtained with ISP’s co-operation, subpoenas, …)
- Similarly for ethernet (MAC address) and other protocols

Why anonymity is difficult?

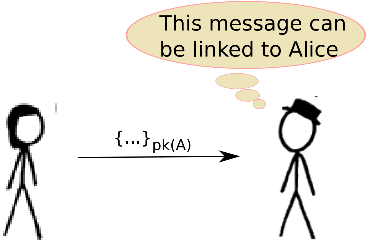
- `wolframalpha.com` : “who am I”?
- Sender’s IP address included in all IP packets
- Already enough to trace someone to ISP/region level
- Can be traced down to individuals using ISP’s logs
  (obtained with ISP’s co-operation, subpoenas, …)
- Similarly for ethernet (MAC address) and other protocols
- Identity leakage via other means (eg cookies)

Communication level vs application level

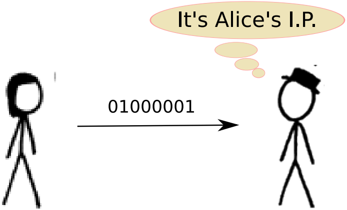
Communication level (our focus)



Application level



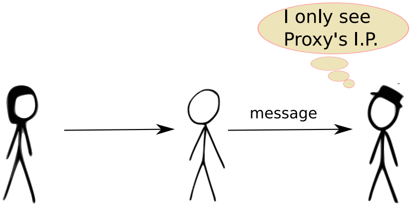
Anonymous communication



How can we approach this problem?

First approach

- Use an anonymous proxy

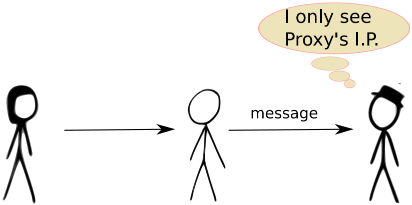


First approach

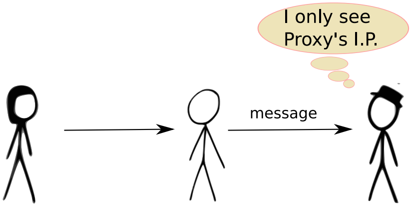
- Use an anonymous proxy

First approach

- Use an anonymous proxy
- Anonymity guarantees



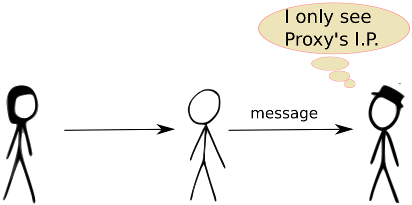
First approach



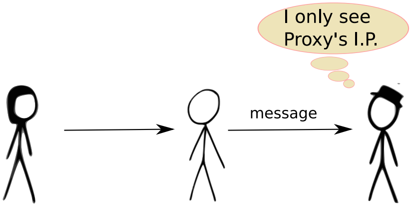
- Use an anonymous proxy
- Anonymity guarantees
  - Sender anonymity, if $A \rightarrow P$ is not visible

First approach

- Use an anonymous proxy
- Anonymity guarantees
  - Sender anonymity, if $A \rightarrow P$ is not visible
  - Receiver anonymity, if $P \rightarrow B$ is not visible

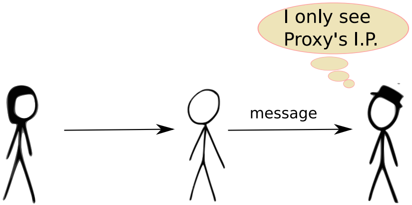


First approach



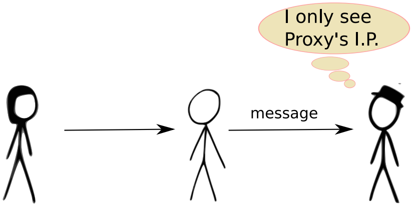
- Use an anonymous proxy
- Anonymity guarantees
  - Sender anonymity, if $A \rightarrow P$ is not visible
  - Receiver anonymity, if $P \rightarrow B$ is not visible
  - If the adversary controls the whole network?

First approach



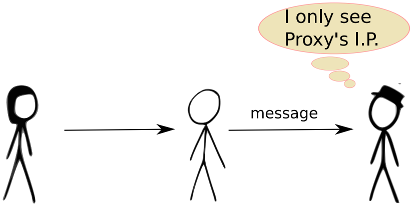
- Use an anonymous proxy
- Anonymity guarantees
  - Sender anonymity, if $A \rightarrow P$ is not visible
  - Receiver anonymity, if $P \rightarrow B$ is not visible
  - If the adversary controls the whole network?

First approach



- Use an anonymous proxy
- Anonymity guarantees
  - Sender anonymity, if $A \rightarrow P$ is not visible
  - Receiver anonymity, if $P \rightarrow B$ is not visible
  - If the adversary controls the whole network?
- Problems

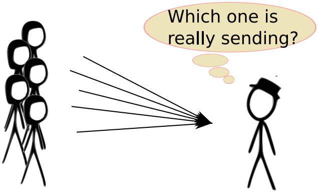
First approach



- Use an anonymous proxy
- Anonymity guarantees
  - Sender anonymity, if $A \rightarrow P$ is not visible
  - Receiver anonymity, if $P \rightarrow B$ is not visible
  - If the adversary controls the whole network?
- Problems
  - We need to trust the proxy
  - Easy to block

Approaches without a trusted party

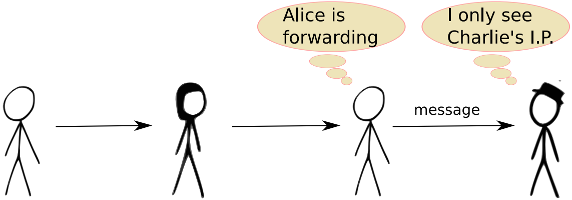
1. Hide message in other traffic



- Alice's traffic should look indistinguishable from everyone else's
- Possible to achieve "strong" anonymity
  - Dining Cryptographers protocol
- But too costly in bandwidth

Approaches without a trusted party

2. Forward message through other users



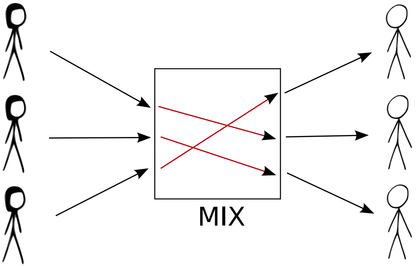
- More efficient
- But challening to deal with an adversary controlling the whole network
- Mixes and Onion routing protocols

Mix neworks

- Stronger adversary
  - Controls the whole network
- But weaker property
  - Hide only the link between a sender and a receiver

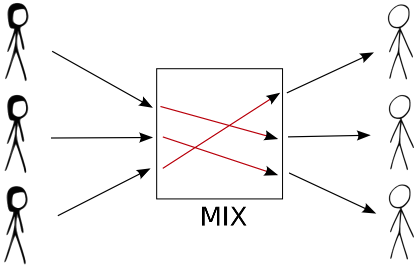
Mix

- Users send messages to the Mix
- The Mix waits until a certain number of messages is received
- Then outputs the messages in some order that is independent from the incoming order (eg random)



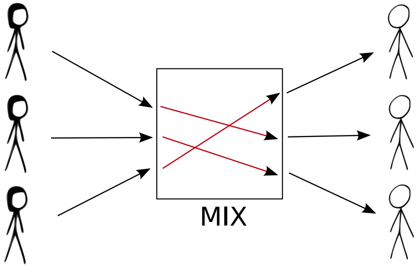
Mix

- Users send messages to the Mix
- The Mix waits until a certain number of messages is received
- Then outputs the messages in some order that is independent from the incoming order (eg random)
- What can a global adversary infer?



Mix

- Users send messages to the Mix
- The Mix waits until a certain number of messages is received
- Then outputs the messages in some order that is independent from the incoming order (eg random)
- What can a global adversary infer?
    - Protect the link between the sender and the receiver



Mix

First goal: bitwise unlinkability

- The input should be indistinguishable from the output

Mix

First goal: bitwise unlinkability

- The input should be indistinguishable from the output
- Encrypt, same sizes

Mix

First goal: bitwise unlinkability

- The input should be indistinguishable from the output
- Encrypt, same sizes
- Prevent against tagging attacks

Mix

Second goal: resistance to traffic analysis

- the order of messages (timing) or other meta-data should not allow to link the sender and receiver

Mix

Second goal: resistance to traffic analysis

- the order of messages (timing) or other meta-data should not allow to link the sender and receiver

Mixing strategies:

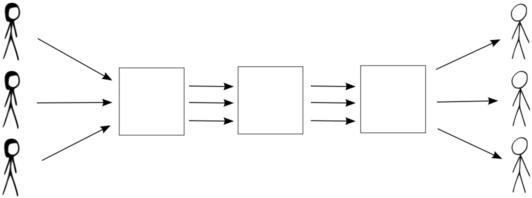
- Threshold Mix: receive N messages, output them in random order
- Pool Mix: keep a pool of M messages. Receive N messages, output N out of $N + M$
- Insufficient traffic $\Rightarrow$ generate dummy messages

Mix: first problem

- We have to trust the Mix

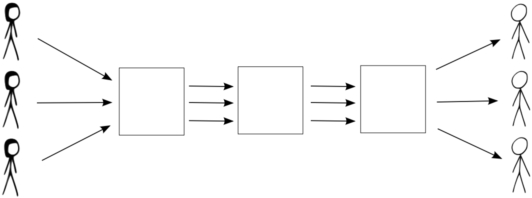
Mix: first problem

- We have to trust the Mix
- Solution: multiple mixes



Mix: first problem

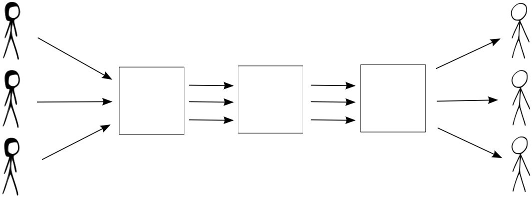
- We have to trust the Mix
- Solution: multiple mixes
- Messages are encrypted with the keys of the mixes in reverse order



Mix: first problem

Various approaches:

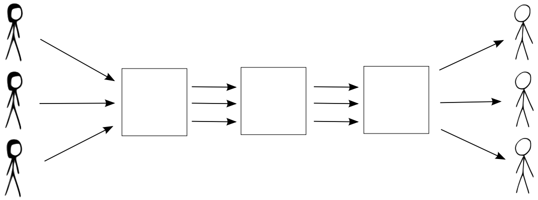
- Cascade mixes: messages pass through all mixes in fixed order
  - A single honest Mix is enough



Mix: first problem

Various approaches:

- Cascade mixes: messages pass through all mixes in fixed order
  - A single honest Mix is enough
- Free routing: mixes are fully connected, messages are routed through random paths
  - Less anonymity, better load balancing



Mix: anonymity analysis

- Does the Mix provide ”strong” sender-receiver unlinkability?
- Adversary goal
  - Distinguish ($A \rightarrow C$, $B \rightarrow D$)
  - From ($A \rightarrow D$, $B \rightarrow C$)

Mix: anonymity analysis

- Does the Mix provide ”strong” sender-receiver unlinkability?
- Adversary goal
  - Distinguish (A $\rightarrow$ C, B $\rightarrow$ D)
  - From (A $\rightarrow$ D, B $\rightarrow$ C)
- What if A is a known friend of C?

Mix: anonymity analysis

- Does the Mix provide ”strong” sender-receiver unlinkability?
- Adversary goal
  - Distinguish (A $\rightarrow$ C, B $\rightarrow$ D)
  - From (A $\rightarrow$ D, B $\rightarrow$ C)
- What if A is a known friend of C?
  - This also reveals information about B and D!

Mix: anonymity analysis

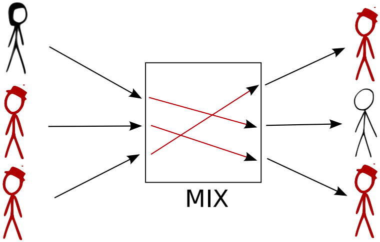
- Does the Mix provide ”strong” sender-receiver unlinkability?
- Adversary goal
  - Distinguish ($A \rightarrow C$, $B \rightarrow D$)
  - From ($A \rightarrow D$, $B \rightarrow C$)
- What if A is a known friend of C?
  - This also reveals information about B and D!
- Anonymity depends on the behaviour of the other users
  - (prior knowledge)

Mix: anonymity analysis

- Extreme case: $(n-1)$ attack
  - the attacker blocks all senders except Alice
  - waits until the mix is flushed
  - sends $n-1$ messages of his own
  - recognizes his messages, thus he infers Alice's recipient

Mix: anonymity analysis

- Extreme case: $(n-1)$ attack
  - the attacker blocks all senders except Alice
  - waits until the mix is flushed
  - sends $n-1$ messages of his own
  - recognizes his messages, thus he infers Alice's recipient



Preventing the $n-1$ attack

- Authenication
  - Difficult to accept in an anonymity system

Preventing the $n - 1$ attack

- Authenication
  - Difficult to accept in an anonymity system
- Delaying-expiring messages
  - Random delay is added by each mix
  - Messages have expiration time
  - Harder for the attacker to flush the mix

Preventing the $n-1$ attack

- Authenication
  - Difficult to accept in an anonymity system
- Delaying-expiring messages
  - Random delay is added by each mix
  - Messages have expiration time
  - Harder for the attacker to flush the mix
- Heartbeat traffic
  - The attacker needs to block other users to flush the mix
  - The mix sends a test message to itself on a certain interval
  - If the message is blocked, inject dummy traffic

Inferring patterns

- Repetitive usage creates patterns that can be observed
- Assume a Mix protocol with n users (one of which is Alice)
- All users are honest and select a receiver with uniform probability $1/n$
- On the i-th run we only observe the set R_i of receivers

Inferring patterns

- Repetitive usage creates patterns that can be observed
- Assume a Mix protocol with n users (one of which is Alice)
- All users are honest and select a receiver with uniform probability $1/n$
- On the i-th run we only observe the set R_i of receivers
- Extreme case: Alice always sends messages to the same receiver r
- With high probability: $\bigcap_i R_i = \{r\}$

Inferring patterns

- Now assume that Alice sends messages to a small set of users { Bob, Paul, Tom }
- We can still infer this set with high probability by simply counting the messages
- Alice's friends will have a higher number of received messages

Inferring patterns

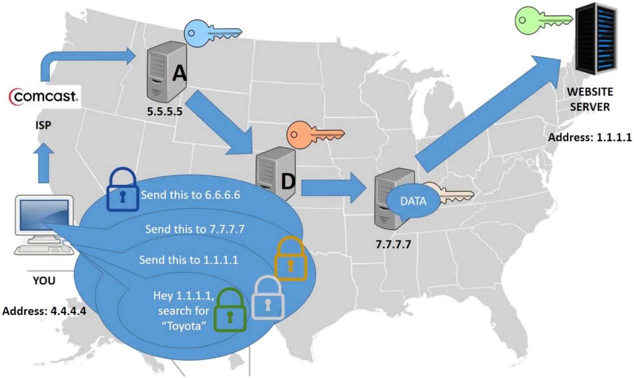
- Now assume that Alice sends messages to a small set of users { Bob, Paul, Tom }
- We can still infer this set with high probability by simply counting the messages
- Alice's friends will have a higher number of received messages
- This probabilistic knowledge can be now used to further de-anonymize other users

Onion routing

- Real-world communication, eg web browsing
- low latency, 1-2 secs round-trip max
- Frequent repeated use
- No time for mixing, delays, etc
- Trade a weaker adversary model for practicality

Onion routing

- Alice selects a short path (3 hops), relays are known
- Encrypt in reverse order (as with mixes)
- Bi-directional channel

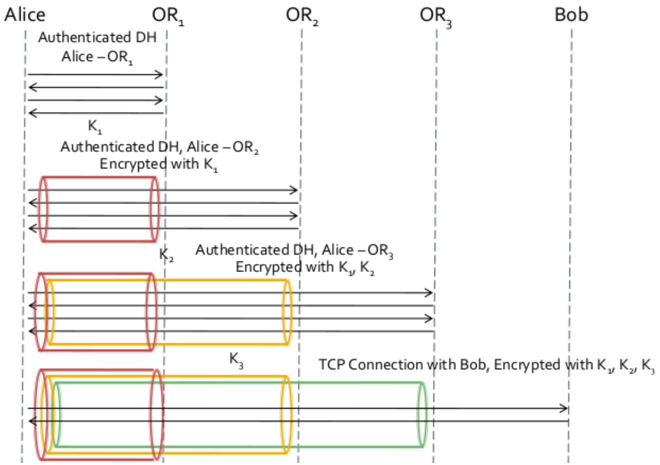


Onion routing

- How can we establish keys with all relays?

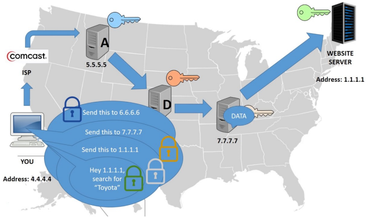
Onion routing

- How can we establish keys with all relays?
- Extend the route via Diffie-Hellman



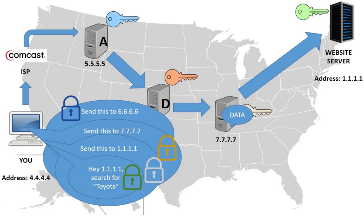
Onion routing, anonymity

- Global adversary?



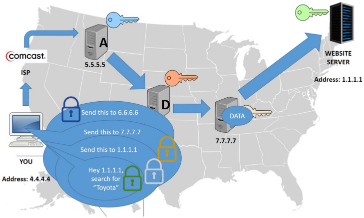
Onion routing, anonymity

- Global adversary? no anonymity
- Partial adversary, controls part of the network/relays



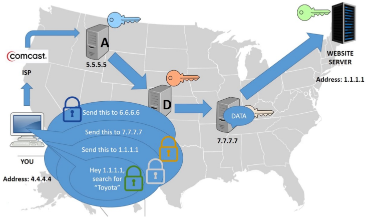
Onion routing, anonymity

- Global adversary? no anonymity
- Partial adversary, controls part of the network/relays
- All nodes controlled : trivial



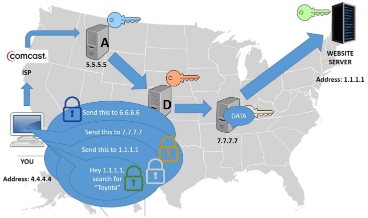
Onion routing, anonymity

- Global adversary? no anonymity
- Partial adversary, controls part of the network/relays
- All nodes controlled : trivial
- Entry & exit nodes controlled : traffic analysis possible



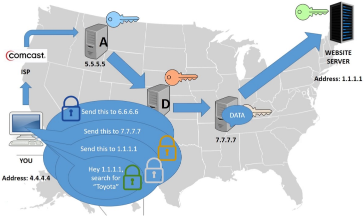
Onion routing, anonymity

- Global adversary? no anonymity
- Partial adversary, controls part of the network/relays
- All nodes controlled : trivial
- Entry & exit nodes controlled : traffic analysis possible
- Attack probability $(\frac{c}{n})^2$



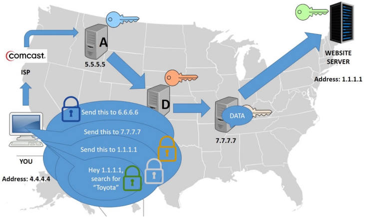
Onion routing, anonymity

- Global adversary? no anonymity
- Partial adversary, controls part of the network/relays
- All nodes controlled : trivial
- Entry & exit nodes controlled : traffic analysis possible
- Attack probability $(\frac{c}{n})^2$
- Useful to have longer routes?



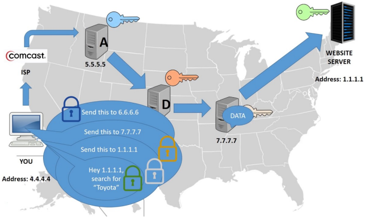
Onion routing, anonymity

- Profiling : detect that Alice communicated with Bob at least once



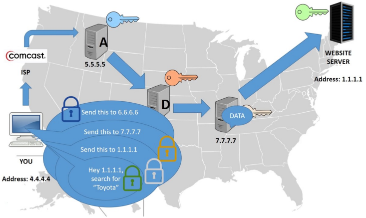
Onion routing, anonymity

- Profiling : detect that Alice communicated with Bob at least once
- Tracing : correlate a specific message



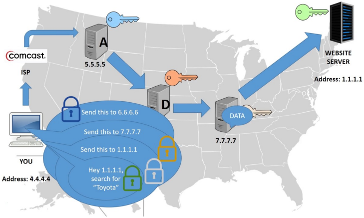
Onion routing, anonymity

- Profiling : detect that Alice communicated with Bob at least once
- Tracing : correlate a specific message
- Long term probability of being profiled :



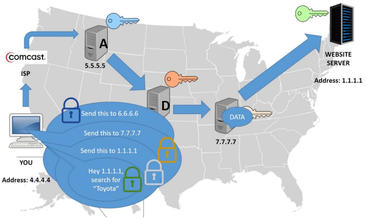
Onion routing, anonymity

- Profiling : detect that Alice communicated with Bob at least once
- Tracing : correlate a specific message
- Long term probability of being profiled : 1 (if entry node changes)



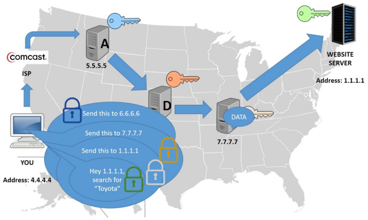
Onion routing, anonymity

- Profiling : detect that Alice communicated with Bob at least once
- Tracing : correlate a specific message
- Long term probability of being profiled : 1 (if entry node changes)
- Solution : fixed entry guard



Onion routing, anonymity

- Profiling : detect that Alice communicated with Bob at least once
- Tracing : correlate a specific message
- Long term probability of being profiled : 1 (if entry node changes)
- Solution : fixed entry guard
  - if honest, profiling/tracing never happens
  - if compromised, higher chances of being traced $\frac{c}{n}$



Onion routing, other problems

- Easy to block

Onion routing, other problems

- Easy to block
- Exit node sees traffic

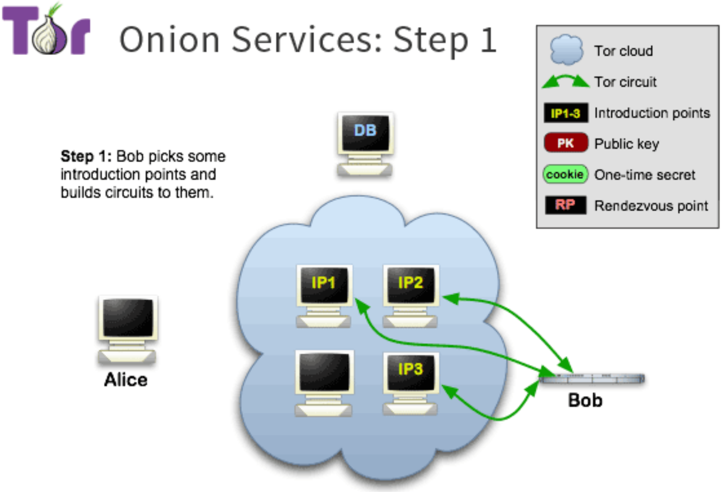
Onion routing, other problems

- Easy to block
- Exit node sees traffic
- Exit node might be identified with illegal behaviour

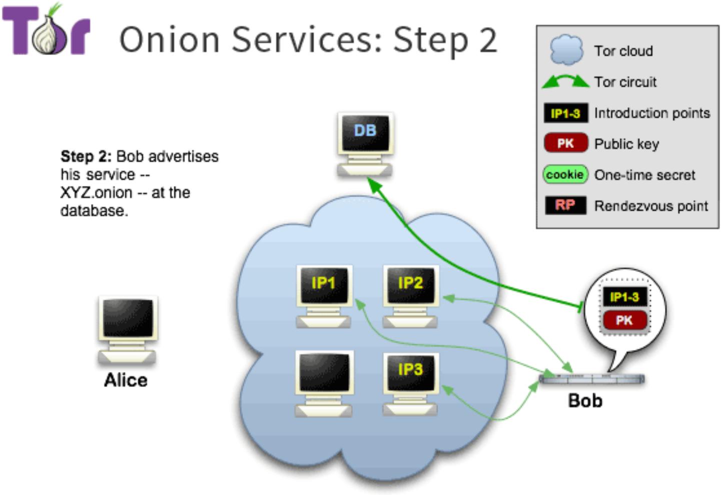
Onion routing, other problems

- Easy to block
- Exit node sees traffic
- Exit node might be identified with illegal behaviour
- No anonymity is provided to the server
  - Solution: onion services

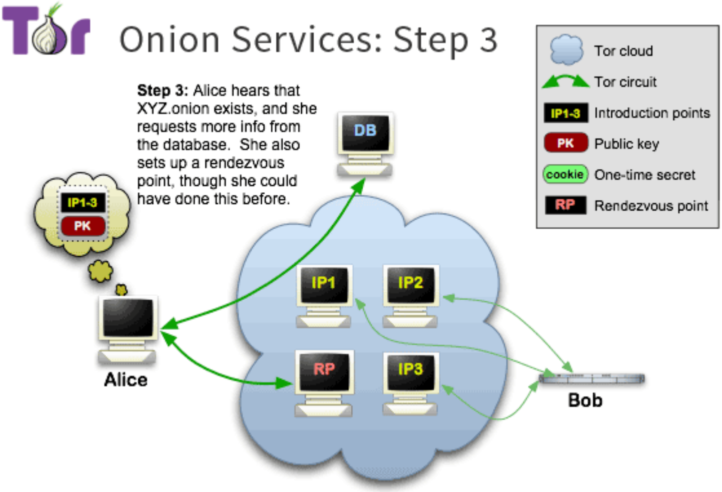
Onion services



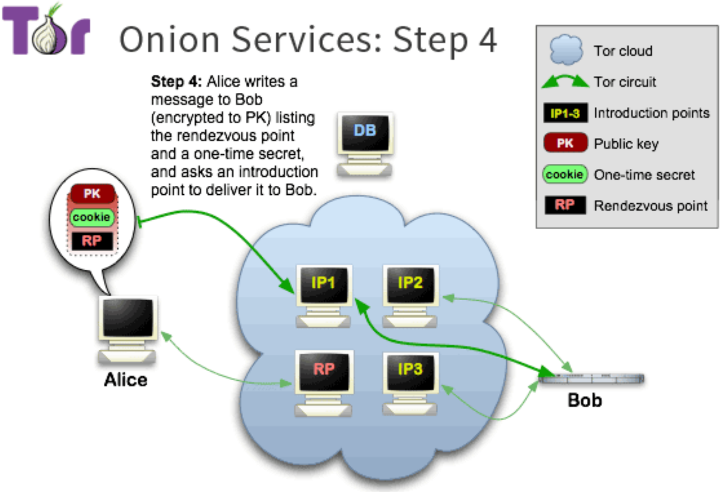
Onion services



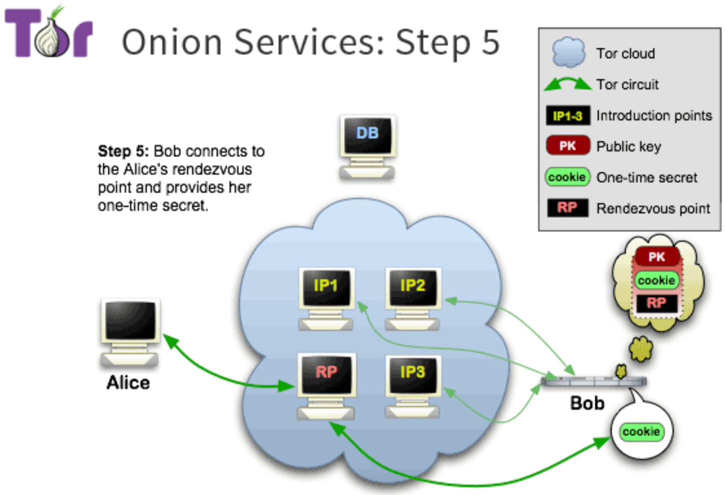
Onion services



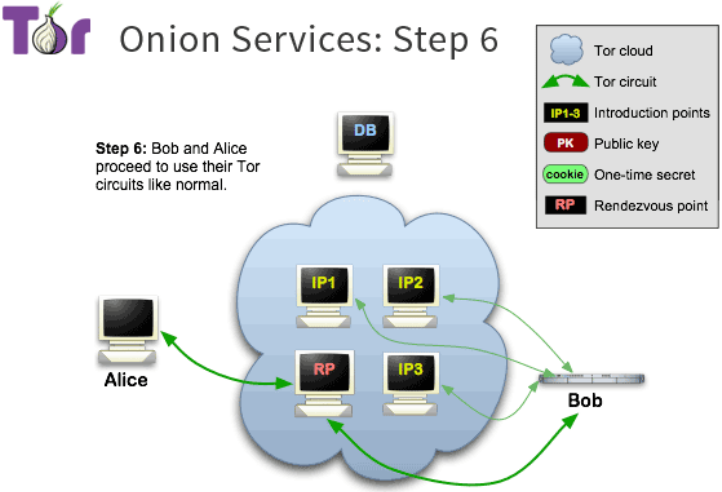
Onion services



Onion services



Onion services



Onion services

- Eg.
  - BBC: `https://www.bbcnewsv2vjtpsuy.onion/`
  - DuckDuckGo: `http://3g2upl4pq6kufc4m.onion/`
  - Facebook: `https://www.facebookcorewwwi.onion/`
  - Riseup: `http://vww6ybal4bd7szmgncyruucpgfkqahzddi37ktceo3ah7ngmcopnpyyd.onion`
- Accessible via the Tor browser