Network Security

Traffic analysis and anonymization

Radboud University Nijmegen, The Netherlands

Autumn 2014

The next weeks

- **Today (Monday, Oct. 13):** last lecture
- **Friday Oct. 17:** werkcollege
- **Monday Oct. 20:** extra werkcollege
- **Friday Oct. 24:** last homework solution + exam Q&A
- **Monday Nov. 3, 12:30–15:30:** Exam

TUN/TAP example

```
tyrion # echo 1 > /proc/sys/net/ipv4/ip_forward
tyrion # ip tuntap add dev tun3 mode tun
tyrion # ip addr add dev tun3 10.0.5.1/24
tyrion # ip l set dev tun3 up

arya # echo 1 > /proc/sys/net/ipv4/ip_forward
arya # ip tuntap add dev tun5 mode tun
arya # ip addr add dev tun5 10.0.5.2/24
arya # ip l set dev tun5 up

tyrion # ssh -o Tunnel=point-to-point -w 3:5 arya
tyrion # ping 10.0.5.2
```

TUN/TAP example

```
tyrion # echo 1 > /proc/sys/net/ipv4/ip_forward
tyrion # ip tuntap add dev tap3 mode tap
tyrion # ip addr add dev tap3 10.0.5.1/24
tyrion # ip l set dev tap3 up

arya # echo 1 > /proc/sys/net/ipv4/ip_forward
arya # ip tuntap add dev tap5 mode tap
arya # ip addr add dev tap5 10.0.5.2/24
arya # ip l set dev tap5 up

tyrion # ssh -o Tunnel=ethernet -w 3:5 arya
tyrion # ping 10.0.5.2
```

A short recap

- Best protection against active and passive attackers: encrypt and authenticate all traffic

A short recap

- Best protection against active and passive attackers: encrypt and authenticate all traffic
- Different security for encryption on different layers:

A short recap

- Best protection against active and passive attackers: encrypt and authenticate all traffic
- Different security for encryption on different layers:
  - Link-layer: protection between two network "neighbors" (example: WPA2)

A short recap

- Best protection against active and passive attackers: encrypt and authenticate all traffic
- Different security for encryption on different layers:
  - Link-layer: protection between two network "neighbors" (example: WPA2)
  - Network layer: encryption between two nodes (example: IPsec)

A short recap

- Best protection against active and passive attackers: encrypt and authenticate all traffic
- Different security for encryption on different layers:
  - Link-layer: protection between two network "neighbors" (example: WPA2)
  - Network layer: encryption between two nodes (example: IPsec)
  - Transport layer: encryption between client and server process (example: SSL/TLS)

A short recap

- Best protection against active and passive attackers: encrypt and authenticate all traffic
- Different security for encryption on different layers:
  - Link-layer: protection between two network "neighbors" (example: WPA2)
  - Network layer: encryption between two nodes (example: IPsec)
  - Transport layer: encryption between client and server process (example: SSL/TLS)
  - Application layer: end-to-end encryption between applications (example: PGP)

A short recap

- Best protection against active and passive attackers: encrypt and authenticate all traffic
- Different security for encryption on different layers:
  - Link-layer: protection between two network "neighbors" (example: WPA2)
  - Network layer: encryption between two nodes (example: IPsec)
  - Transport layer: encryption between client and server process (example: SSL/TLS)
  - Application layer: end-to-end encryption between applications (example: PGP)
- IPsec has two modes of operation: transport and tunneling

A short recap

- Best protection against active and passive attackers: encrypt and authenticate all traffic
- Different security for encryption on different layers:
  - Link-layer: protection between two network "neighbors" (example: WPA2)
  - Network layer: encryption between two nodes (example: IPsec)
  - Transport layer: encryption between client and server process (example: SSL/TLS)
  - Application layer: end-to-end encryption between applications (example: PGP)
- IPsec has two modes of operation: transport and tunneling
- IPsec has two main protocols: authentication headers (AH) and encapsulating security payloads (ESP)

A short recap

- Best protection against active and passive attackers: encrypt and authenticate all traffic
- Different security for encryption on different layers:
  - Link-layer: protection between two network "neighbors" (example: WPA2)
  - Network layer: encryption between two nodes (example: IPsec)
  - Transport layer: encryption between client and server process (example: SSL/TLS)
  - Application layer: end-to-end encryption between applications (example: PGP)
- IPsec has two modes of operation: transport and tunneling
- IPsec has two main protocols: authentication headers (AH) and encapsulating security payloads (ESP)
- IPsec's Security Associations (SA) establish unidirectional security relations

A short recap

- Best protection against active and passive attackers: encrypt and authenticate all traffic
- Different security for encryption on different layers:
  - Link-layer: protection between two network "neighbors" (example: WPA2)
  - Network layer: encryption between two nodes (example: IPsec)
  - Transport layer: encryption between client and server process (example: SSL/TLS)
  - Application layer: end-to-end encryption between applications (example: PGP)
- IPsec has two modes of operation: transport and tunneling
- IPsec has two main protocols: authentication headers (AH) and encapsulating security payloads (ESP)
- IPsec's Security Associations (SA) establish unidirectional security relations
- TLS uses (long) handshake to agree on cipher suites, establish session keys

A short recap

- Best protection against active and passive attackers: encrypt and authenticate all traffic
- Different security for encryption on different layers:
  - Link-layer: protection between two network "neighbors" (example: WPA2)
  - Network layer: encryption between two nodes (example: IPsec)
  - Transport layer: encryption between client and server process (example: SSL/TLS)
  - Application layer: end-to-end encryption between applications (example: PGP)
- IPsec has two modes of operation: transport and tunneling
- IPsec has two main protocols: authentication headers (AH) and encapsulating security payloads (ESP)
- IPsec's Security Associations (SA) establish unidirectional security relations
- TLS uses (long) handshake to agree on cipher suites, establish session keys
- All cipher suites in TLS have (or had) some problems

A short recap

- Best protection against active and passive attackers: encrypt and authenticate all traffic
- Different security for encryption on different layers:
  - Link-layer: protection between two network "neighbors" (example: WPA2)
  - Network layer: encryption between two nodes (example: IPsec)
  - Transport layer: encryption between client and server process (example: SSL/TLS)
  - Application layer: end-to-end encryption between applications (example: PGP)
- IPsec has two modes of operation: transport and tunneling
- IPsec has two main protocols: authentication headers (AH) and encapsulating security payloads (ESP)
- IPsec's Security Associations (SA) establish unidirectional security relations
- TLS uses (long) handshake to agree on cipher suites, establish session keys
- All cipher suites in TLS have (or had) some problems
- Best option: TLS_ECDHE_RSA_WITH_AES256_GCM_SHA384

Who do you trust?

- HTTPS (HTTP over SSL/TLS) uses pre-installed root certificates in the browser
- Operating systems come with various pre-installed certificates
- Authenticating a communication partner means: follow chain of trust to root CA

Who do you trust?

- HTTPS (HTTP over SSL/TLS) uses pre-installed root certificates in the browser
- Operating systems come with various pre-installed certificates
- Authenticating a communication partner means: follow chain of trust to root CA
- Compromise one root CA and all browsers are compromised
- Forge a root CA's certificate and all browsers are compromised

Who do you trust?

- HTTPS (HTTP over SSL/TLS) uses pre-installed root certificates in the browser
- Operating systems come with various pre-installed certificates
- Authenticating a communication partner means: follow chain of trust to root CA
- Compromise one root CA and all browsers are compromised
- Forge a root CA's certificate and all browsers are compromised
- Rogue CA certificate from MD5 vulnerabilities, 2008: http://www.win.tue.nl/hashclash/rogue-ca/

Who do you trust?

- HTTPS (HTTP over SSL/TLS) uses pre-installed root certificates in the browser
- Operating systems come with various pre-installed certificates
- Authenticating a communication partner means: follow chain of trust to root CA
- Compromise one root CA and all browsers are compromised
- Forge a root CA's certificate and all browsers are compromised
- Rogue CA certificate from MD5 vulnerabilities, 2008: http://www.win.tue.nl/hashclash/rogue-ca/
- DigiNotar compromised in 2011: >300,000 Iranian Gmail users compromised

OpenSSL Heartbleed Bug

Bug in the implementation of the Heartbeat Extension (RFC 6520):

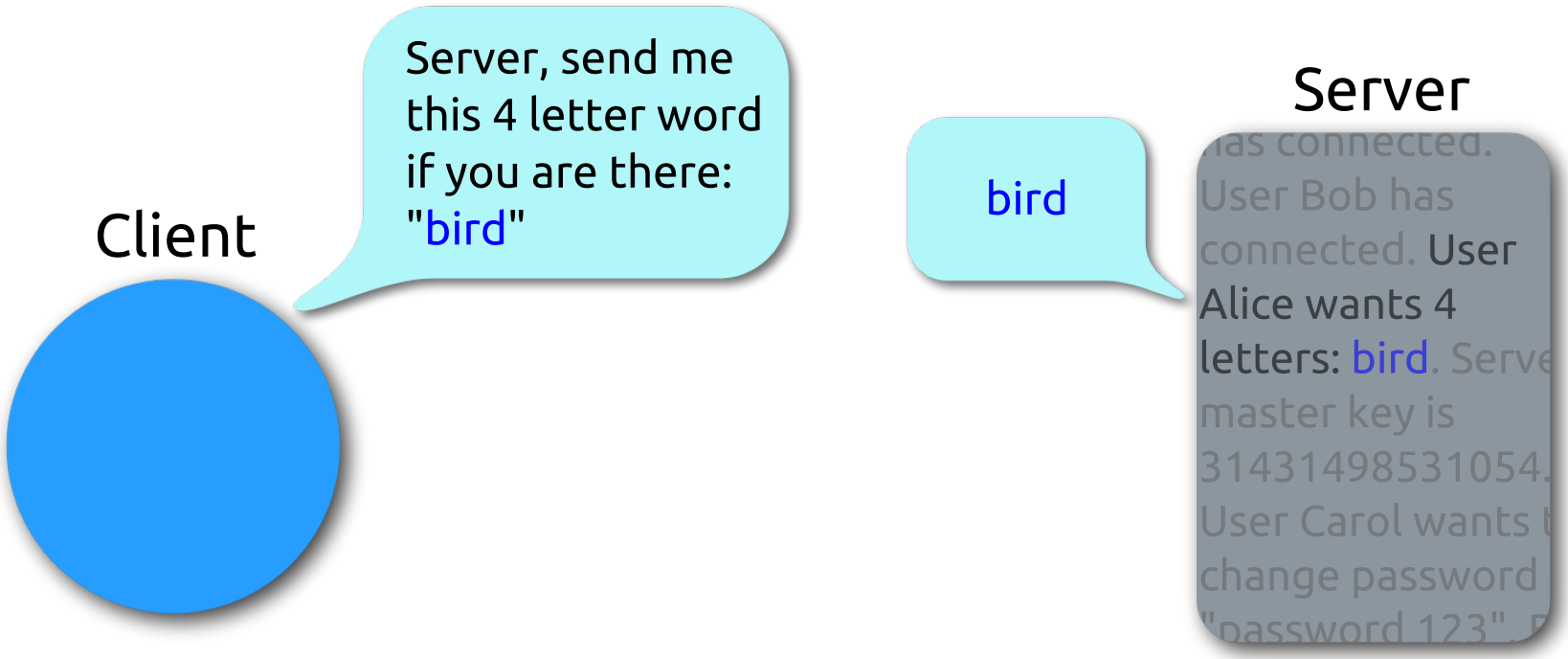
```
struct {
   HeartbeatMessageType type;
   uint16 payload_length;
   opaque payload[HeartbeatMessage.payload_length];
   opaque padding[padding_length];
 } HeartbeatMessage;

 [...]
 When a HeartbeatRequest message is received [...],
 the receiver MUST send a corresponding HeartbeatResponse
 message carrying an exact copy of the payload of the received
 HeartbeatRequest.
```

OpenSSL failed to check actual length of payload data.

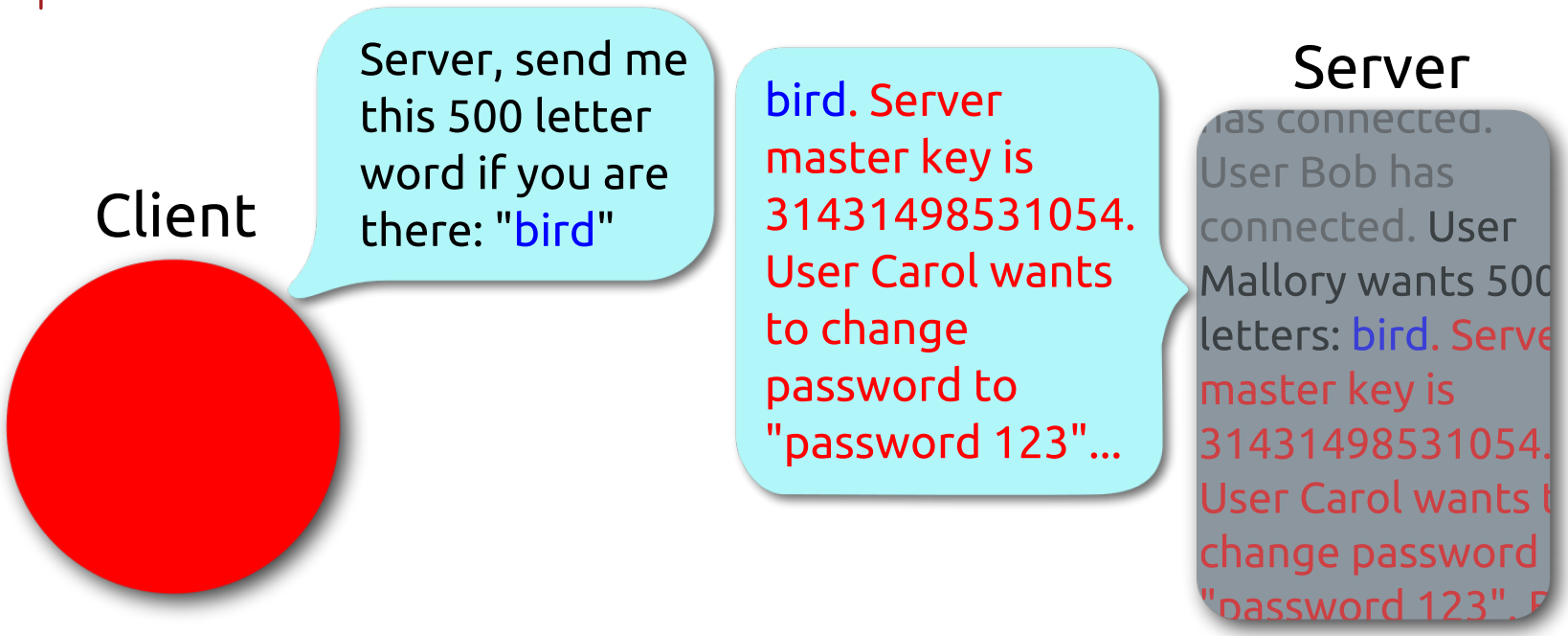
OpenSSL Heartbleed Bug

Heartbeat – Normal usage



OpenSSL Heartbleed Bug

Heartbeat – Malicious usage



SSLstrip I

- Consider SSL/TLS for HTTP (i.e., HTTPS):
  - Attacker cannot read data (because they are encrypted)
  - Attacker cannot modify data (because they are authenticated)
- Target of the attacker: avoid HTTPS

SSLstrip I

- Consider SSL/TLS for HTTP (i.e., HTTPS):
  - Attacker cannot read data (because they are encrypted)
  - Attacker cannot modify data (because they are authenticated)
- Target of the attacker: avoid HTTPS
- Most users visit websites by clicking on links (or HTTP 301/302)
- Consider a website `http://www.target.com` (note that it's HTTP!)
- Assume that this website has a link of the following form:

```
<a href="https://www.target.com/login.html">login</a>
```

- User clicks on this link and uses HTTPS

SSLstrip I

- Consider SSL/TLS for HTTP (i.e., HTTPS):
  - Attacker cannot read data (because they are encrypted)
  - Attacker cannot modify data (because they are authenticated)
- Target of the attacker: avoid HTTPS
- Most users visit websites by clicking on links (or HTTP 301/302)
- Consider a website `http://www.target.com` (note that it's HTTP!)
- Assume that this website has a link of the following form:

```
<a href="https://www.target.com/login.html">login</a>
```

- User clicks on this link and uses HTTPS
- Active MitM attacker can modify this link to

```
<a href="http://www.target.com/login.html">login</a>
```

- User clicks on this link and uses HTTP!

SSLstrip II

- This attack was presented by Moxie Marlinspike in 2009
- Automated tool to perform this attack: `sslstrip`
- More information:
  http://www.thoughtcrime.org/software/sslstrip/

SSLstrip II

- This attack was presented by Moxie Marlinspike in 2009
- Automated tool to perform this attack: `sslstrip`
- More information:
  http://www.thoughtcrime.org/software/sslstrip/
- User *can* notice this attack (by looking at browser's address bar)
- Current browsers don't show huge differences for HTTP vs. HTTPS
- Reasonable assumption that most users won't notice all of the time

SSLstrip II

- This attack was presented by Moxie Marlinspike in 2009
- Automated tool to perform this attack: `sslstrip`
- More information:
  http://www.thoughtcrime.org/software/sslstrip/
- User *can* notice this attack (by looking at browser's address bar)
- Current browsers don't show huge differences for HTTP vs. HTTPS
- Reasonable assumption that most users won't notice all of the time
- HTTP Strict Transport Security (HSTS) attempts to solve the problem
- Web server declares that browsers have to use HTTPS

SSLstrip II

- This attack was presented by Moxie Marlinspike in 2009
- Automated tool to perform this attack: `sslstrip`
- More information:
  http://www.thoughtcrime.org/software/sslstrip/
- User *can* notice this attack (by looking at browser's address bar)
- Current browsers don't show huge differences for HTTP vs. HTTPS
- Reasonable assumption that most users won't notice all of the time
- HTTP Strict Transport Security (HSTS) attempts to solve the problem
- Web server declares that browsers have to use HTTPS
- Attacker can strip the HSTS header in the *first request* to the server
- Firefox and Chrome ship with a list of known HSTS sites

How much web traffic is encrypted?

How much web traffic is encrypted?

How much web traffic is encrypted?

From the article:

"Early last year–before the Snowden revelations–encrypted traffic accounted for 2.29 percent of all peak hour traffic in North America, according to Sandvine's report. Now, it spans 3.8 percent. But that's a small jump compared to other parts of the world. In Europe, encrypted traffic went from 1.47 percent to 6.10 percent, and in Latin America, it increased from 1.8 percent to 10.37 percent."

—Klint Finley on wired.com, May 16, 2014.

The perfect world (crypto-wise)

Imagine a world in which ...

- ... all Internet traffic is encrypted and authenticated,

The perfect world (crypto-wise)

Imagine a world in which ...

- ... all Internet traffic is encrypted and authenticated,
- ... e-mails are all PGP encrypted and signed,

The perfect world (crypto-wise)

Imagine a world in which ...

- ... all Internet traffic is encrypted and authenticated,
- ... e-mails are all PGP encrypted and signed,
- ... everybody is using cipher suites that offer high security,

The perfect world (crypto-wise)

Imagine a world in which ...

- ... all Internet traffic is encrypted and authenticated,
- ... e-mails are all PGP encrypted and signed,
- ... everybody is using cipher suites that offer high security,
- ... all trusted parties are trustworthy,

The perfect world (crypto-wise)

Imagine a world in which ...

- ... all Internet traffic is encrypted and authenticated,
- ... e-mails are all PGP encrypted and signed,
- ... everybody is using cipher suites that offer high security,
- ... all trusted parties are trustworthy,
- ... crypto implementations are correct and secure,

The perfect world (crypto-wise)

Imagine a world in which ...

- ... all Internet traffic is encrypted and authenticated,
- ... e-mails are all PGP encrypted and signed,
- ... everybody is using cipher suites that offer high security,
- ... all trusted parties are trustworthy,
- ... crypto implementations are correct and secure,
- ... applied cryptographers have trouble finding a job.

What *does* an attacker see?

EU's Data Retention Directive

"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."

What *does* an attacker see?

EU's Data Retention Directive

"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."

From Article 5:

- data necessary to trace and identify the source of a communication
- data necessary to identify the destination of a communication

What *does* an attacker see?

EU's Data Retention Directive

"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."

From Article 5:

- data necessary to trace and identify the source of a communication
- data necessary to identify the destination of a communication
- data necessary to identify the date, time and duration of a communication

What *does* an attacker see?

EU's Data Retention Directive

"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."

From Article 5:

- data necessary to trace and identify the source of a communication
- data necessary to identify the destination of a communication
- data necessary to identify the date, time and duration of a communication
- data necessary to identify the type of communication

What *does* an attacker see?

EU's Data Retention Directive

"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."

From Article 5:

- data necessary to trace and identify the source of a communication
- data necessary to identify the destination of a communication
- data necessary to identify the date, time and duration of a communication
- data necessary to identify the type of communication
- data necessary to identify users' communication equipment or what purports to be their equipment

What *does* an attacker see?

EU's Data Retention Directive

"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."

From Article 5:

- data necessary to trace and identify the source of a communication
- data necessary to identify the destination of a communication
- data necessary to identify the date, time and duration of a communication
- data necessary to identify the type of communication
- data necessary to identify users' communication equipment or what purports to be their equipment
- data necessary to identify the location of mobile communication equipment

What *does* an attacker see?

EU's Data Retention Directive

"Member States shall ensure that the categories of data specified in Article 5 are retained for periods of not less than six months and not more than two years from the date of the communication."

From Article 5:

- data necessary to trace and identify the source of a communication
- data necessary to identify the destination of a communication
- data necessary to identify the date, time and duration of a communication
- data necessary to identify the type of communication
- data necessary to identify users' communication equipment or what purports to be their equipment
- data necessary to identify the location of mobile communication equipment

Encrypting and authenticating content does not prevent any of this!

What can you do with “meta data”?

“Metadata absolutely tells you everything about somebody’s life. If you have enough metadata you don’t really need content. . . [It’s] sort of embarrassing how predictable we are as human beings.”

—Stewart Baker, former general counsel of the NSA

What can you do with “meta data”?

“Metadata absolutely tells you everything about somebody’s life. If you have enough metadata you don’t really need content. . . [It’s] sort of embarrassing how predictable we are as human beings.”

—Stewart Baker, former general counsel of the NSA

“We kill people based on metadata.”

—Michael Hayden, former director of the NSA and the CIA

Is "metadata" all an attacker gets?

- Common assumption: an attacker sees only traffic data ("meta data")
- Example, interview with Jimmy Wales (Wikipedia founder):

  "***You've said that you're going to start encrypting communications on Wikipedia as a result***...

  We have done. It's not completely finished yet but the only thing that GCHQ, hopefully, can see is that you're looking at Wikipedia. They can't see which article you're reading. It's not the government's business to know what everybody is reading."

Is "metadata" all an attacker gets?

- Common assumption: an attacker sees only traffic data ("meta data")
- Example, interview with Jimmy Wales (Wikipedia founder):

  "***You've said that you're going to start encrypting communications on Wikipedia as a result***...

  We have done. It's not completely finished yet but the only thing that GCHQ, hopefully, can see is that you're looking at Wikipedia. They can't see which article you're reading. It's not the government's business to know what everybody is reading."
- Small experiment: 10 Wikipedia pages, load one at random through HTTPS
- Attacker sniffs the network, tries to figure out which one

Is "metadata" all an attacker gets?

- Common assumption: an attacker sees only traffic data ("meta data")
- Example, interview with Jimmy Wales (Wikipedia founder):

  "***You've said that you're going to start encrypting communications on Wikipedia as a result***...

  We have done. It's not completely finished yet but the only thing that GCHQ, hopefully, can see is that you're looking at Wikipedia. They can't see which article you're reading. It's not the government's business to know what everybody is reading."
- Small experiment: 10 Wikipedia pages, load one at random through HTTPS
- Attacker sniffs the network, tries to figure out which one

Is "metadata" all an attacker gets?

- Common assumption: an attacker sees only traffic data ("meta data")
- Example, interview with Jimmy Wales (Wikipedia founder):

  "***You've said that you're going to start encrypting communications on Wikipedia as a result***...

  We have done. It's not completely finished yet but the only thing that GCHQ, hopefully, can see is that you're looking at Wikipedia. They can't see which article you're reading. It's not the government's business to know what everybody is reading."
- Small experiment: 10 Wikipedia pages, load one at random through HTTPS
- Attacker sniffs the network, tries to figure out which one

Is "metadata" all an attacker gets?

- Common assumption: an attacker sees only traffic data ("meta data")
- Example, interview with Jimmy Wales (Wikipedia founder):

  "***You've said that you're going to start encrypting communications on Wikipedia as a result***...

  We have done. It's not completely finished yet but the only thing that GCHQ, hopefully, can see is that you're looking at Wikipedia. They can't see which article you're reading. It's not the government's business to know what everybody is reading."
- Small experiment: 10 Wikipedia pages, load one at random through HTTPS
- Attacker sniffs the network, tries to figure out which one
- Not that hard:
  - 10 webpages have different amount of pictures
  - Browser sends one request per image
  - Attacker counts requests, distinguishes websites

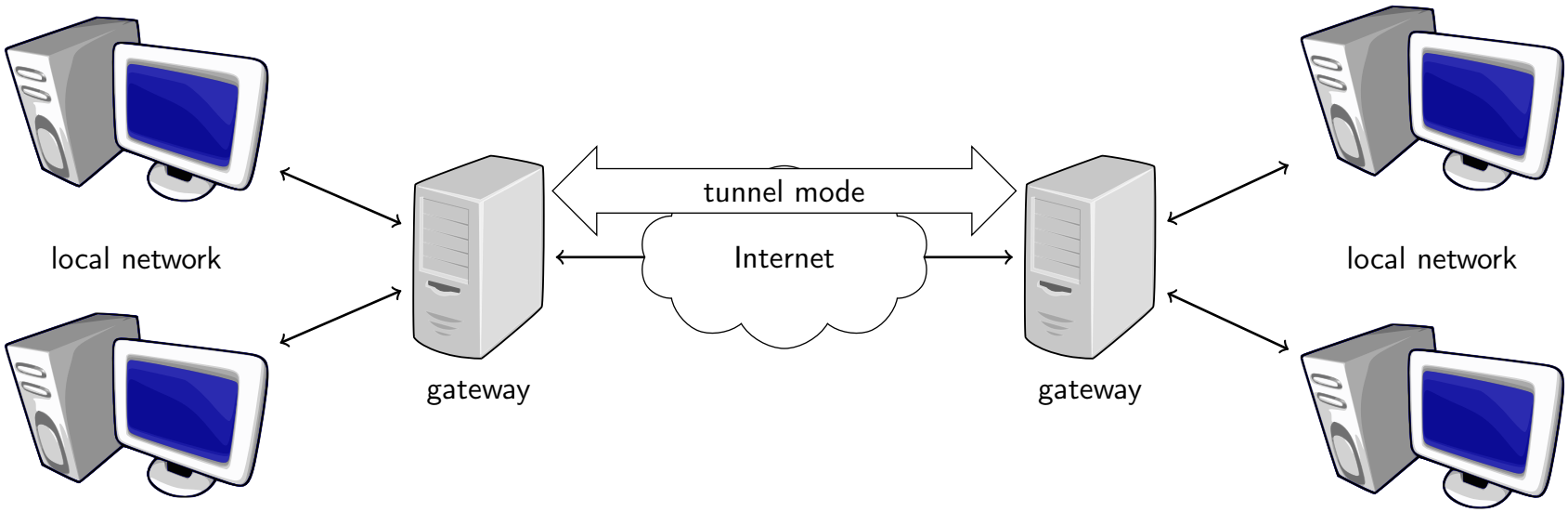
Is "metadata" all an attacker gets?

- Common assumption: an attacker sees only traffic data ("meta data")
- Example, interview with Jimmy Wales (Wikipedia founder):

  "***You've said that you're going to start encrypting communications on Wikipedia as a result***...

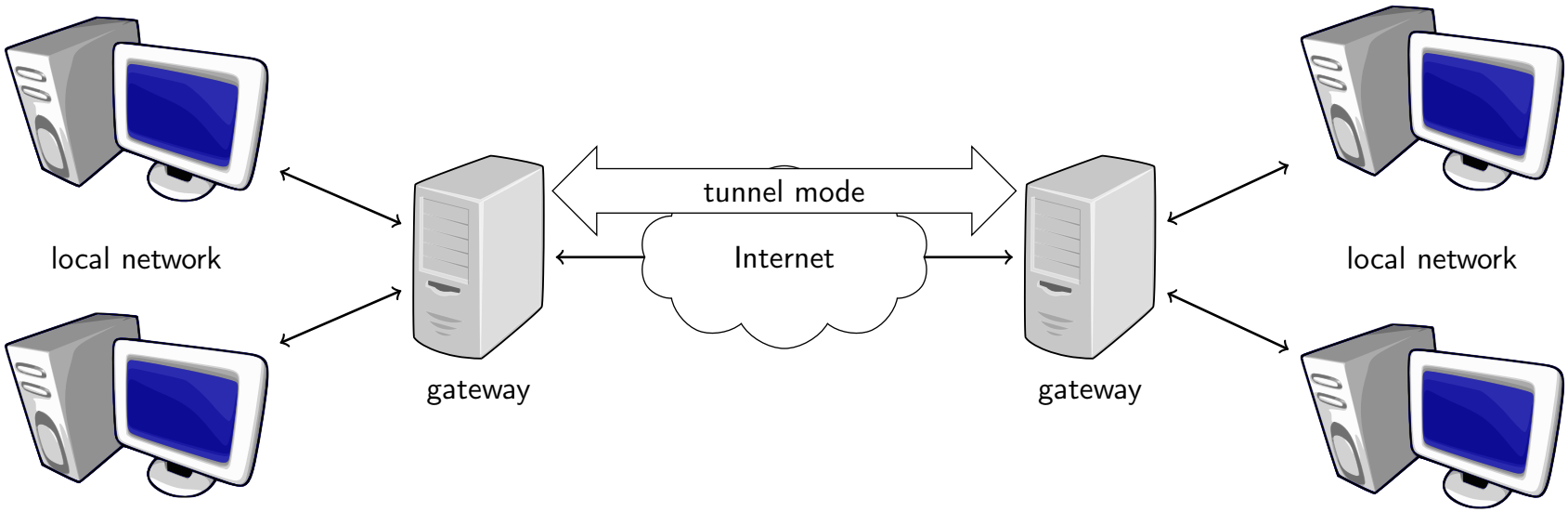
  We have done. It's not completely finished yet but the only thing that GCHQ, hopefully, can see is that you're looking at Wikipedia. They can't see which article you're reading. It's not the government's business to know what everybody is reading."
- Small experiment: 10 Wikipedia pages, load one at random through HTTPS
- Attacker sniffs the network, tries to figure out which one
- Not that hard:
  - 10 webpages have different amount of pictures
  - Browser sends one request per image
  - Attacker counts requests, distinguishes websites
  - This is not the only thing an attacker sees; more in the homework

IPsec ESP in tunnel mode



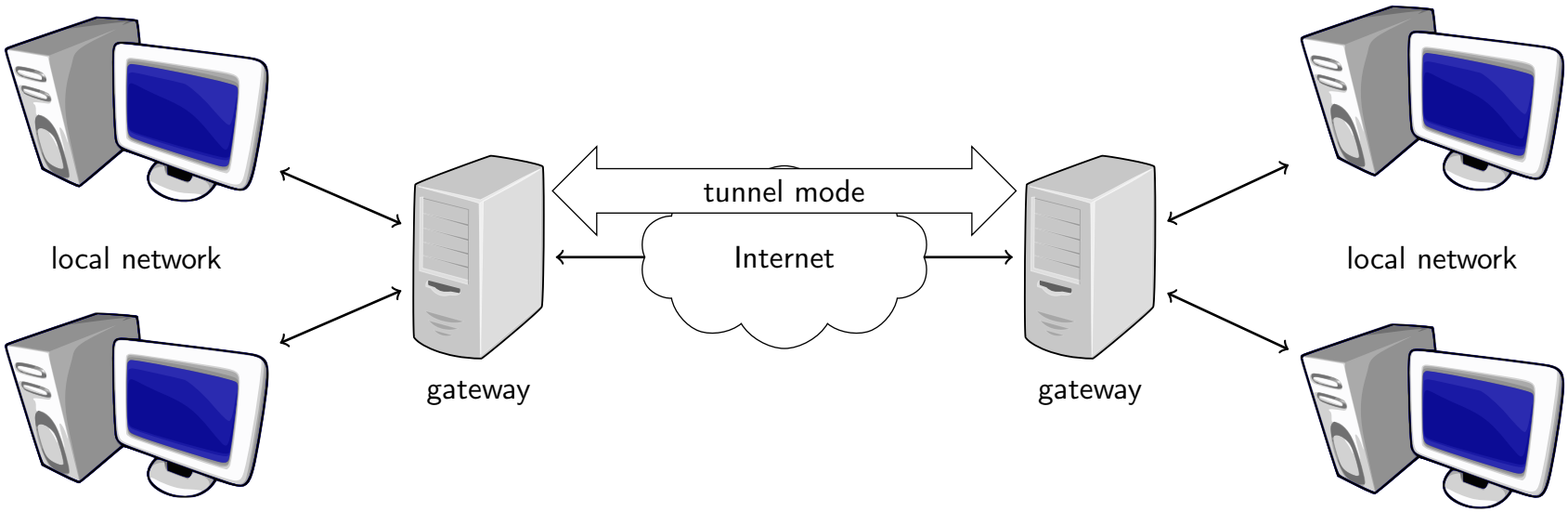
- Everything between the gateways has the *gateways' addresses*
- Nodes behind the gateways are indistinguishable
- RFC 2406 calls this "limited traffic flow confidentiality"

IPsec ESP in tunnel mode



- Everything between the gateways has the *gateways' addresses*
- Nodes behind the gateways are indistinguishable
- RFC 2406 calls this "limited traffic flow confidentiality"
- Problem 1: Does not help against state-level attacker who can request gateway's logfiles

IPsec ESP in tunnel mode



- Everything between the gateways has the *gateways' addresses*
- Nodes behind the gateways are indistinguishable
- RFC 2406 calls this "limited traffic flow confidentiality"
- Problem 1: Does not help against state-level attacker who can request gateway's logfiles
- Problem 2: Potentially small *anonymity set*

Anonymizing proxies

- Somewhat similar idea (without crypto): use a proxy server
- Typically: application-specific proxies (e.g., HTTP proxies)
- Requests to websites come from proxy
- All users behind the proxy are indistinguishable
- Various problems:
  1. Single point of failure against state-level attackers

Anonymizing proxies

- Somewhat similar idea (without crypto): use a proxy server
- Typically: application-specific proxies (e.g., HTTP proxies)
- Requests to websites come from proxy
- All users behind the proxy are indistinguishable
- Various problems:
  1. Single point of failure against state-level attackers
  2. Proxy somewhere in the Internet: easy to correlate ingoing/outgoing traffic

Anonymizing proxies

- Somewhat similar idea (without crypto): use a proxy server
- Typically: application-specific proxies (e.g., HTTP proxies)
- Requests to websites come from proxy
- All users behind the proxy are indistinguishable
- Various problems:
  1. Single point of failure against state-level attackers
  2. Proxy somewhere in the Internet: easy to correlate ingoing/outgoing traffic
  3. No crypto protection to the proxy

Anonymizing proxies

- Somewhat similar idea (without crypto): use a proxy server
- Typically: application-specific proxies (e.g., HTTP proxies)
- Requests to websites come from proxy
- All users behind the proxy are indistinguishable
- Various problems:
  1. Single point of failure against state-level attackers
  2. Proxy somewhere in the Internet: easy to correlate ingoing/outgoing traffic
  3. No crypto protection to the proxy
- Can add crypto to the proxy (e.g., OpenVPN Service)
- That still does not solve problems 1 and 2

Mix Networks

- Idea for anonymous electronic mail by Chaum, 1981: mixing networks
- Assume that Alice want to anonymously send message M to Bob

Mix Networks

- Idea for anonymous electronic mail by Chaum, 1981: mixing networks
- Assume that Alice want to anonymously send message M to Bob
- Uses intermediate computer called *mix* and public keys
  - K_B of Bob, and
  - K_M of the mix

Mix Networks

- Idea for anonymous electronic mail by Chaum, 1981: mixing networks
- Assume that Alice want to anonymously send message M to Bob
- Uses intermediate computer called *mix* and public keys
  - K_B of Bob, and
  - K_M of the mix
- Sends to mix: $K_M(R_1, K_B(R_0, M))$ for random R_0, R_1

Mix Networks

- Idea for anonymous electronic mail by Chaum, 1981: mixing networks
- Assume that Alice want to anonymously send message M to Bob
- Uses intermediate computer called *mix* and public keys
  - K_B of Bob, and
  - K_M of the mix
- Sends to mix: $K_M(R_1, K_B(R_0, M))$ for random R_0, R_1
- Mix collects many such mails, decrypts to $K_B(R_0, M)$

Mix Networks

- Idea for anonymous electronic mail by Chaum, 1981: mixing networks
- Assume that Alice want to anonymously send message M to Bob
- Uses intermediate computer called *mix* and public keys
  - K_B of Bob, and
  - K_M of the mix
- Sends to mix: $K_M(R_1, K_B(R_0, M))$ for random R_0, R_1
- Mix collects many such mails, decrypts to $K_B(R_0, M)$
- Sends mails in lexicographic order to receivers
- Receiver Bob decrypts and obtains M

Mix Networks

- Idea for anonymous electronic mail by Chaum, 1981: mixing networks
- Assume that Alice want to anonymously send message M to Bob
- Uses intermediate computer called *mix* and public keys
  - K_B of Bob, and
  - K_M of the mix
- Sends to mix: $K_M(R_1, K_B(R_0, M))$ for random R_0, R_1
- Mix collects many such mails, decrypts to $K_B(R_0, M)$
- Sends mails in lexicographic order to receivers
- Receiver Bob decrypts and obtains M
- Achieves anonymity if encrypted messages are indistinguishable
- Very important: never repeat input and output!
- Has high communication latency (wait for enough messages)

Return Addresses

- Now Alice can send mail to Bob, how about replies?
- Need a way for Bob to reply without revealing Alice's address/identity

Return Addresses

- Now Alice can send mail to Bob, how about replies?
- Need a way for Bob to reply without revealing Alice's address/identity
- Alice includes a return address her message encrypted to Bob:

$$K_M(R_1, A_X), K_X$$

- R_1, K_X are random one-time symmetric keys
- A_X is Alice's real address

Return Addresses

- Now Alice can send mail to Bob, how about replies?
- Need a way for Bob to reply without revealing Alice's address/identity
- Alice includes a return address her message encrypted to Bob:

$$K_M(R_1, A_X), K_X$$

- R_1, K_X are random one-time symmetric keys
- A_X is Alice's real address
- Bob can send response M as

$$K_M(R_1, A_X), K_X(R_0, M)$$

Return Addresses

- Now Alice can send mail to Bob, how about replies?
- Need a way for Bob to reply without revealing Alice's address/identity
- Alice includes a return address her message encrypted to Bob:

$$K_M(R_1, A_X), K_X$$

- R_1, K_X are random one-time symmetric keys
- A_X is Alice's real address
- Bob can send response M as

$$K_M(R_1, A_X), K_X(R_0, M)$$

- Mix receives and recovers R_1, A_X, sends to Alice

$$R_1(K_X(R_0, M))$$

Return Addresses

- Now Alice can send mail to Bob, how about replies?
- Need a way for Bob to reply without revealing Alice's address/identity
- Alice includes a return address her message encrypted to Bob:

$$K_M(R_1, A_X), K_X$$

- R_1, K_X are random one-time symmetric keys
- A_X is Alice's real address
- Bob can send response M as

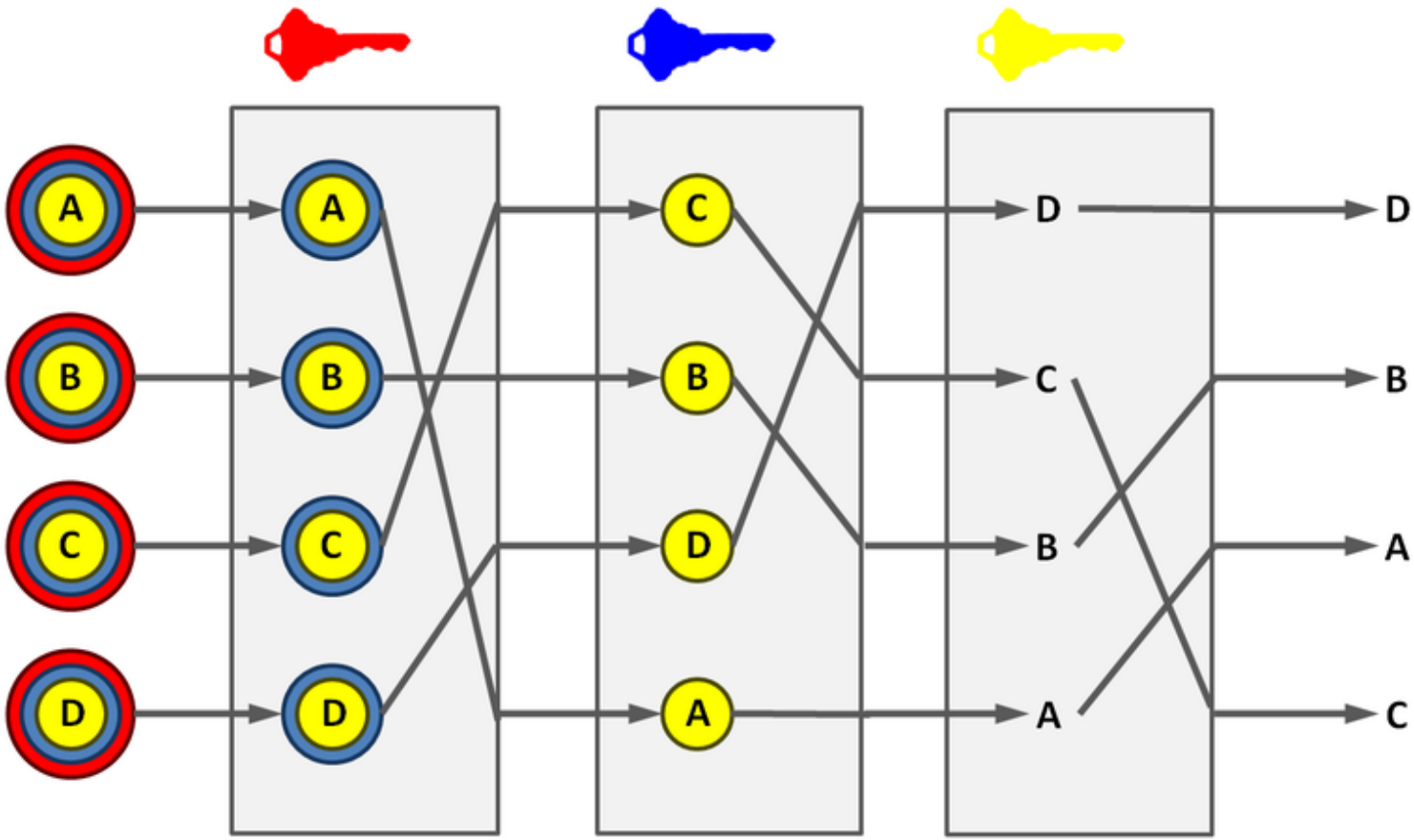
$$K_M(R_1, A_X), K_X(R_0, M)$$

- Mix receives and recovers R_1, A_X, sends to Alice

$$R_1(K_X(R_0, M))$$

- Only Alice can decrypt, because only she knows both K_X and R_1

Cascading Mixes



Mix Nets vs. Anonymizing proxies

Mix Nets

+ No single point of failure (with cascading)
+ Inbound/output-traffic analysis does not de-anonymize
+ Generally good anonymity

Mix Nets vs. Anonymizing proxies

Mix Nets

+ No single point of failure (with cascading)
+ Inbound/output-traffic analysis does not de-anonymize
+ Generally good anonymity
– Slow public-key cryptography (at least in vanilla mix nets)
– Long latency

Mix Nets vs. Anonymizing proxies

Mix Nets

+ No single point of failure (with cascading)
+ Inbound/output-traffic analysis does not de-anonymize
+ Generally good anonymity
– Slow public-key cryptography (at least in vanilla mix nets)
– Long latency

Anon. Proxies

+ Low latency
+ No overhead from slow crypto

Mix Nets vs. Anonymizing proxies

Mix Nets

+ No single point of failure (with cascading)
+ Inbound/output-traffic analysis does not de-anonymize
+ Generally good anonymity
− Slow public-key cryptography (at least in vanilla mix nets)
− Long latency

Anon. Proxies

+ Low latency
+ No overhead from slow crypto
− Single point of failure
− Inbound/output-traffic analysis de-anonymizes
− Fairly weak anonymity

Mix Nets vs. Anonymizing proxies

Mix Nets

+ No single point of failure (with cascading)
+ Inbound/output-traffic analysis does not de-anonymize
+ Generally good anonymity
– Slow public-key cryptography (at least in vanilla mix nets)
– Long latency

Anon. Proxies

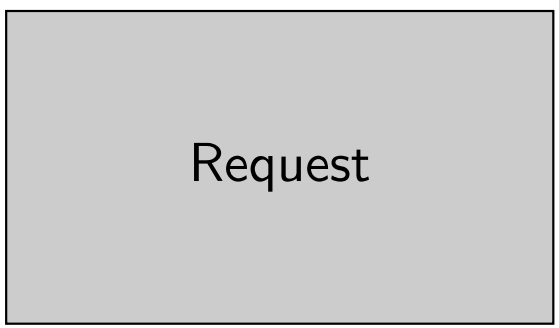
+ Low latency
+ No overhead from slow crypto
– Single point of failure
– Inbound/output-traffic analysis de-anonymizes
– Fairly weak anonymity

Idea of Tor (The Onion Router): Combine advantages:

- Use cascade of "proxies", called *Tor relays* or *Tor nodes*
- Use fast symmetric crypto instead of asymmetric crypto

Onion Routing and Tor

- Assume that user shares symmetric keys with three *relays*:
  - Entry relay R_1 (key K_{R_1})
  - Guard relay R_2 (key K_{R_2})
  - Exit relay R_3 (key K_{R_3})
- Wants to anonymously send request to `www.wikileaks.org`



Onion Routing and Tor

- Assume that user shares symmetric keys with three *relays*:
  - Entry relay R_1 (key K_{R_1})
  - Guard relay R_2 (key K_{R_2})
  - Exit relay R_3 (key K_{R_3})
- Wants to anonymously send request to `www.wikileaks.org`
- Prepares packet as follows:
  - Write dest. `www.wikileaks.org`, encrypt with K_{R_3}

Onion Routing and Tor

- Assume that user shares symmetric keys with three *relays*:
  - Entry relay R_1 (key K_{R_1})
  - Guard relay R_2 (key K_{R_2})
  - Exit relay R_3 (key K_{R_3})
- Wants to anonymously send request to `www.wikileaks.org`
- Prepares packet as follows:
  - Write dest. `www.wikileaks.org`, encrypt with K_{R_3}
  - Write dest. R_3 encrypt with K_{R_2}

Onion Routing and Tor

- Assume that user shares symmetric keys with three *relays*:
  - Entry relay R_1 (key K_{R_1})
  - Guard relay R_2 (key K_{R_2})
  - Exit relay R_3 (key K_{R_3})
- Wants to anonymously send request to `www.wikileaks.org`
- Prepares packet as follows:
  - Write dest. `www.wikileaks.org`, encrypt with K_{R_3}
  - Write dest. R_3 encrypt with K_{R_2}
  - Write dest. R_2 encrypt with K_{R_1}

Onion Routing and Tor

- Assume that user shares symmetric keys with three *relays*:
  - Entry relay R_1 (key K_{R_1})
  - Guard relay R_2 (key K_{R_2})
  - Exit relay R_3 (key K_{R_3})
- Wants to anonymously send request to `www.wikileaks.org`
- Prepares packet as follows:
  - Write dest. `www.wikileaks.org`, encrypt with K_{R_3}
  - Write dest. R_3 encrypt with K_{R_2}
  - Write dest. R_2 encrypt with K_{R_1}
- Send this packet to R_1

Onion Routing and Tor

- R_1 receives packet, removes encryption with K_{R_1}

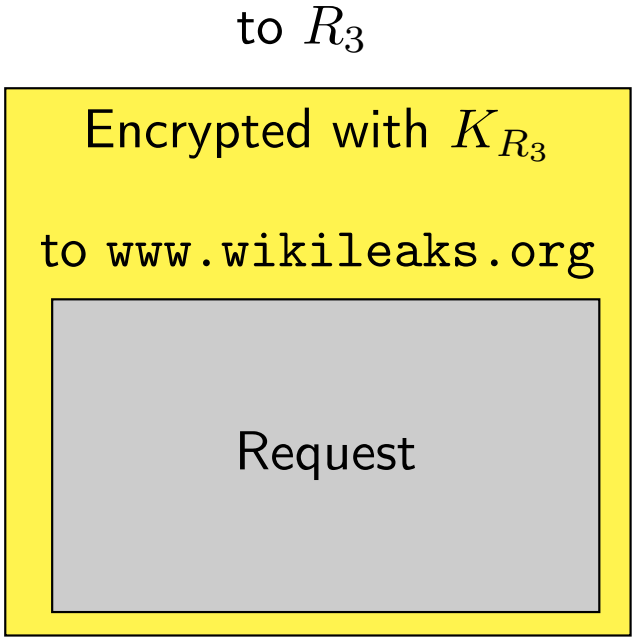
Onion Routing and Tor

- R_1 receives packet, removes encryption with K_{R_1}
- Sees next destination: R_2, forwards

Onion Routing and Tor

- R_1 receives packet, removes encryption with K_{R_1}
- Sees next destination: R_2, forwards
- R_2 receives packet, removes encryption with K_{R_2}

Onion Routing and Tor



- R_1 receives packet, removes encryption with K_{R_1}
- Sees next destination: R_2, forwards
- R_2 receives packet, removes encryption with K_{R_2}
- Sees next destination: R_3, forwards

Onion Routing and Tor

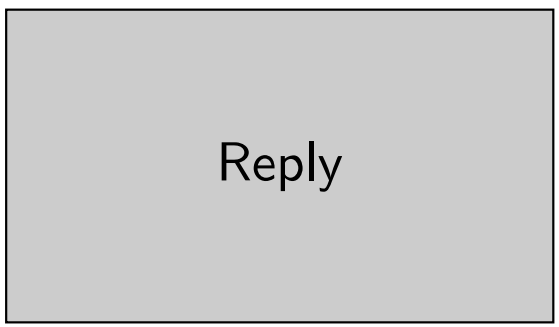
- R_1 receives packet, removes encryption with K_{R_1}
- Sees next destination: R_2, forwards
- R_2 receives packet, removes encryption with K_{R_2}
- Sees next destination: R_3, forwards
- R_3 receives packet, removes encryption with K_{R_3}

Onion Routing and Tor

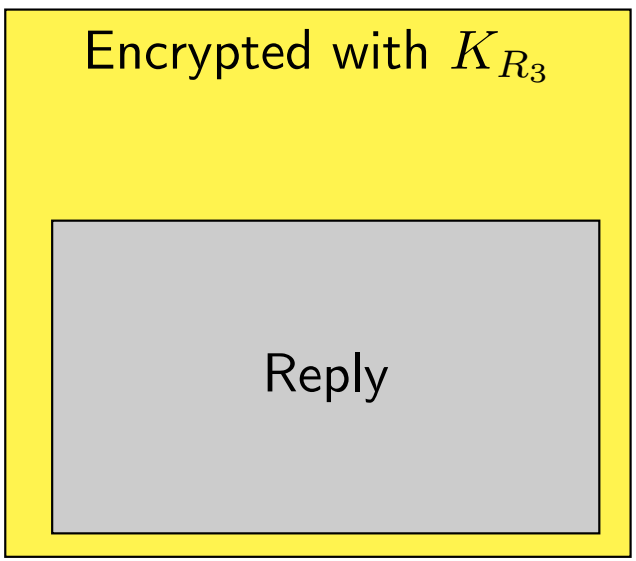
- R_1 receives packet, removes encryption with K_{R_1}
- Sees next destination: R_2, forwards
- R_2 receives packet, removes encryption with K_{R_2}
- Sees next destination: R_3, forwards
- R_3 receives packet, removes encryption with K_{R_3}
- Sees next destination: `www.wikileaks.org`, sends request

Reply from `www.wikileaks.org`

- R_3 receives reply from `www.wikileaks.org`

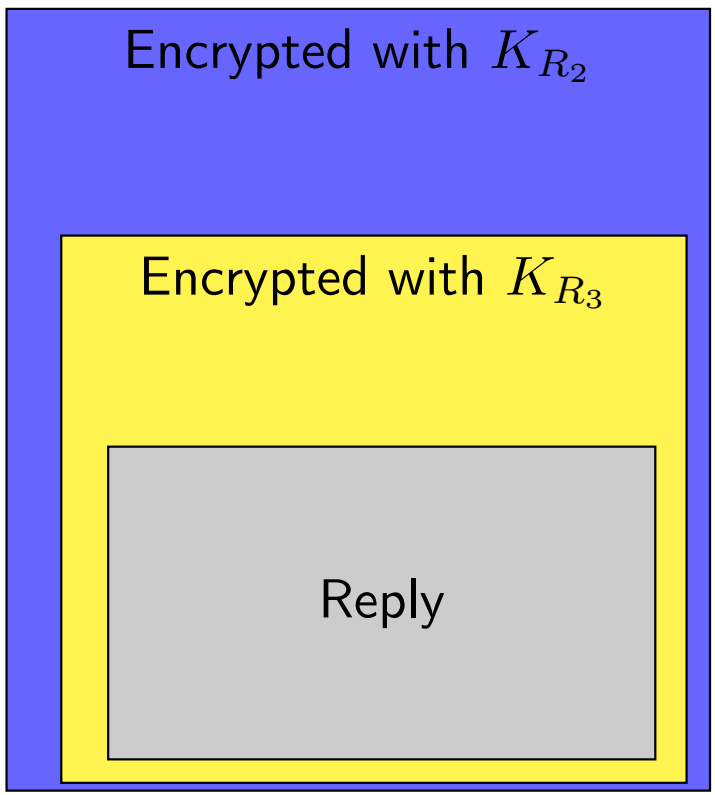


Reply from `www.wikileaks.org`



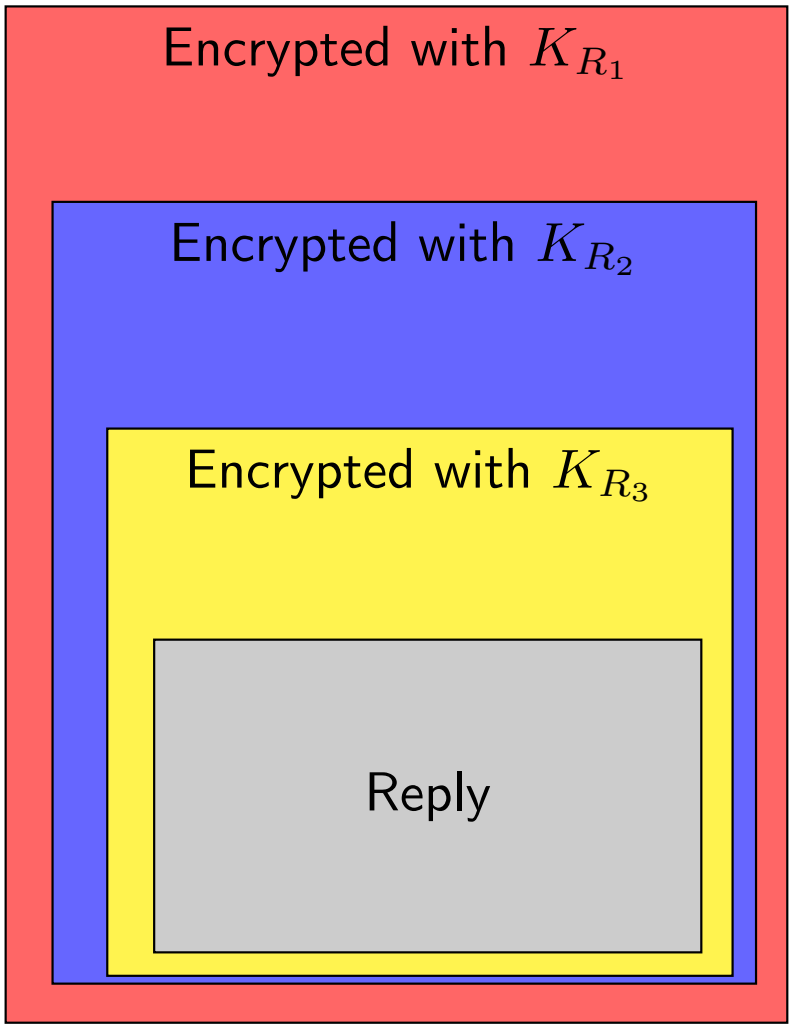
- R_3 receives reply from `www.wikileaks.org`
- R_3 encrypts with K_{R_3}, sends to R_2

Reply from `www.wikileaks.org`



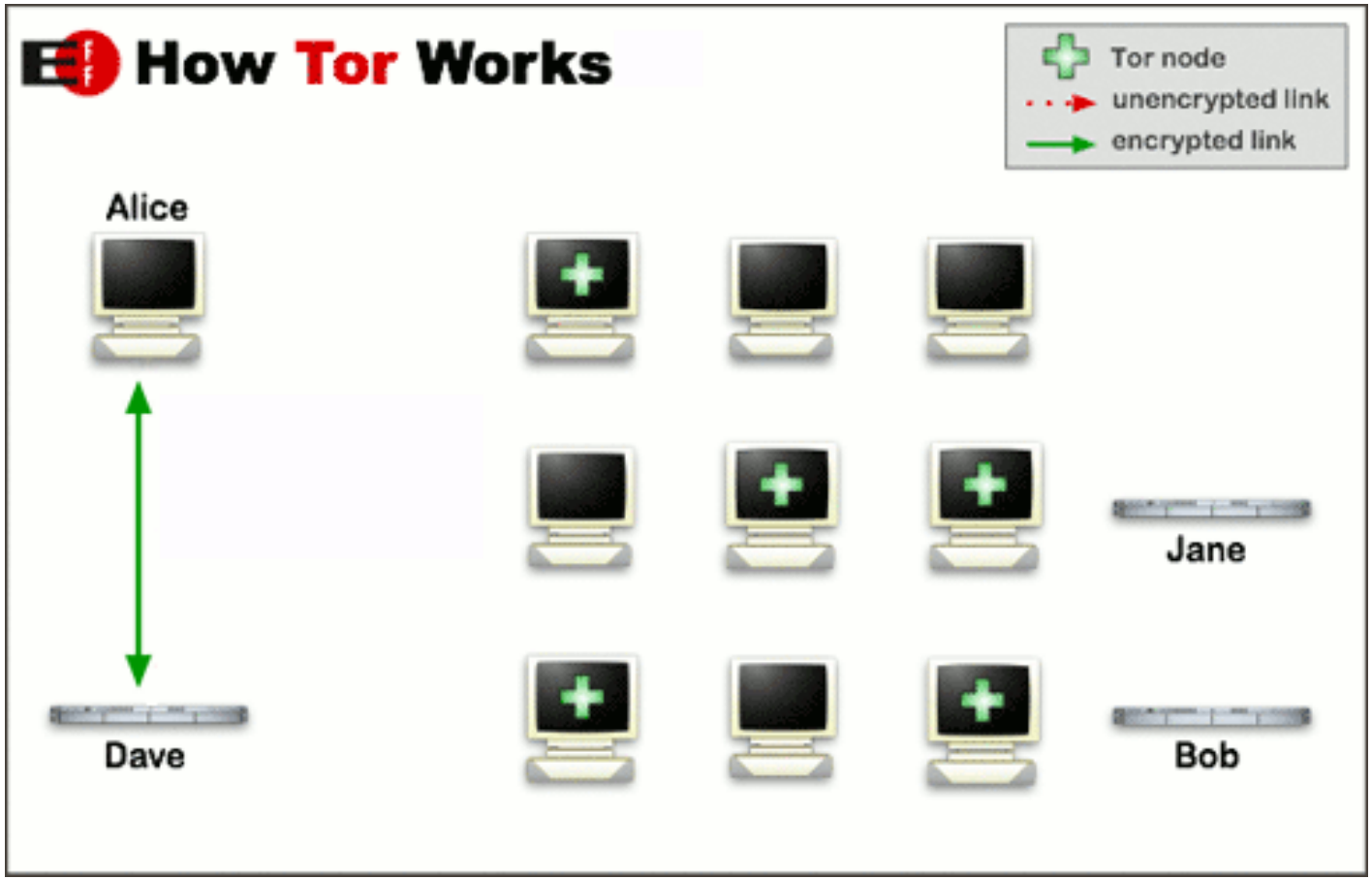
- R_3 receives reply from `www.wikileaks.org`
- R_3 encrypts with K_{R_3}, sends to R_2
- R_2 encrypts with K_{R_2}, sends to R_1

Reply from `www.wikileaks.org`



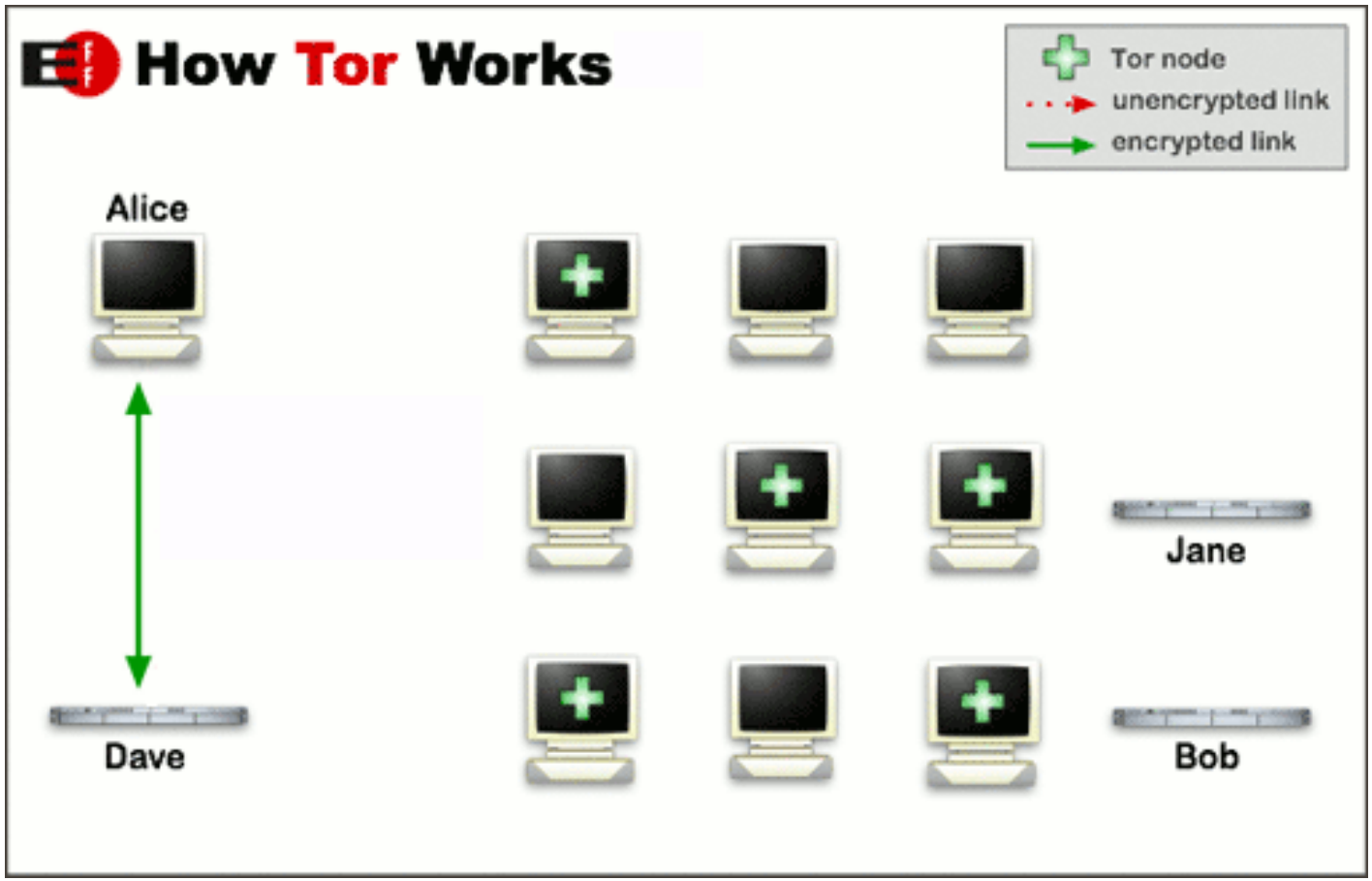
- R_3 receives reply from `www.wikileaks.org`
- R_3 encrypts with K_{R_3}, sends to R_2
- R_2 encrypts with K_{R_2}, sends to R_1
- R_1 encrypts with K_{R_1}, sends to Tor client

Establishing a Circuit



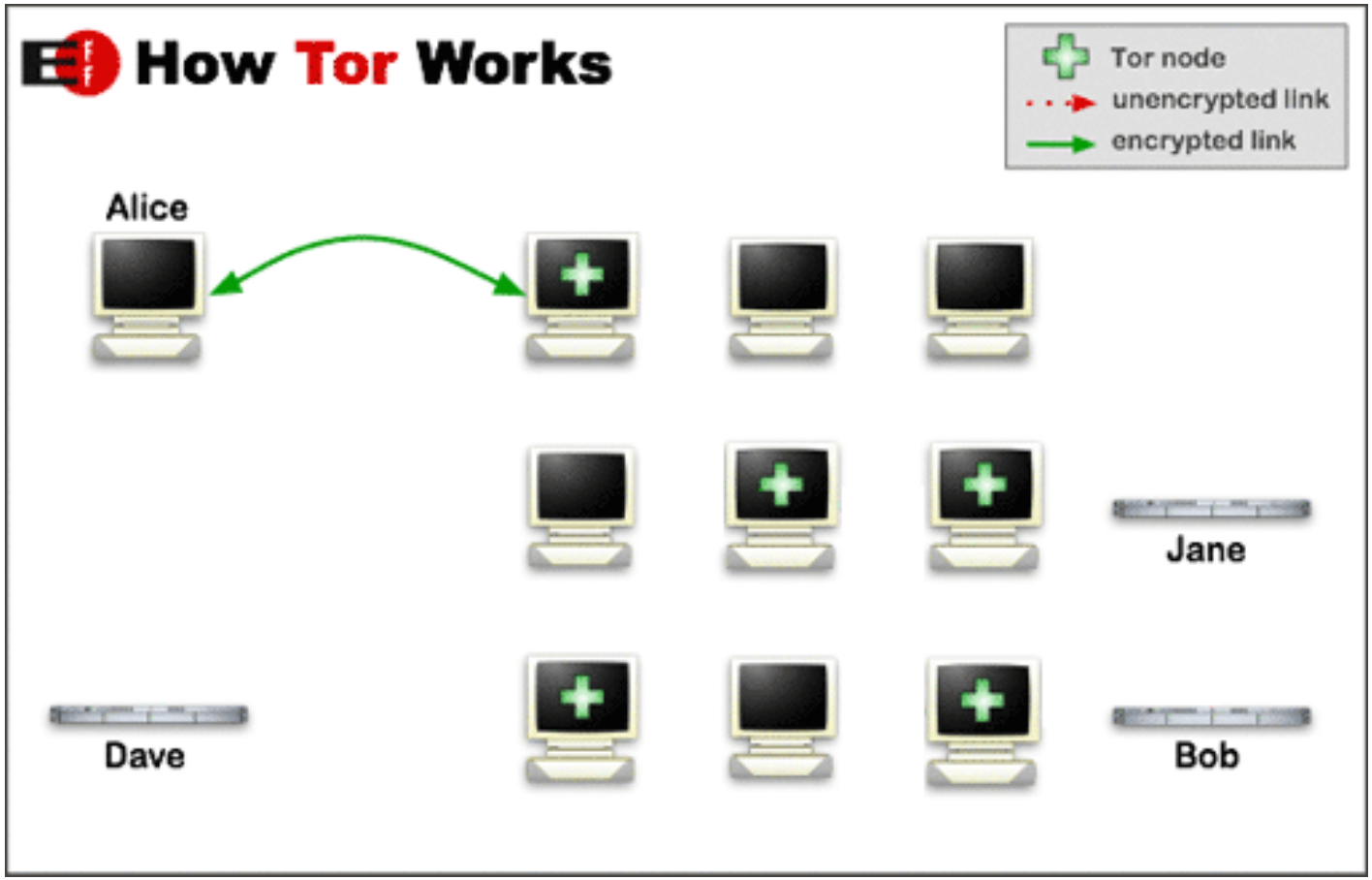
Request listing of Tor nodes from directory server (DS)

Establishing a Circuit



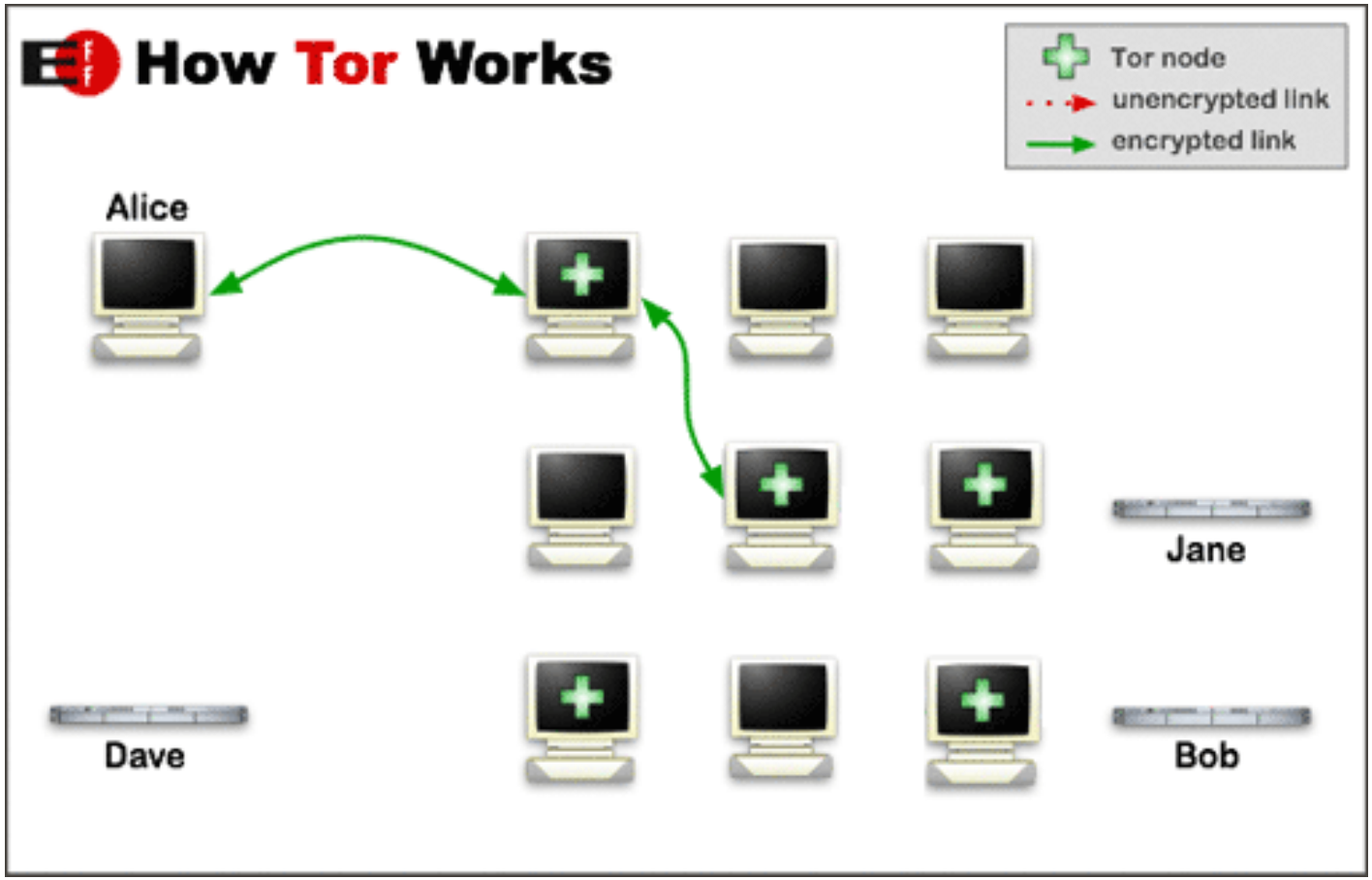
Pick entry, guard, and exit node; obtain their public keys from DS

Establishing a Circuit



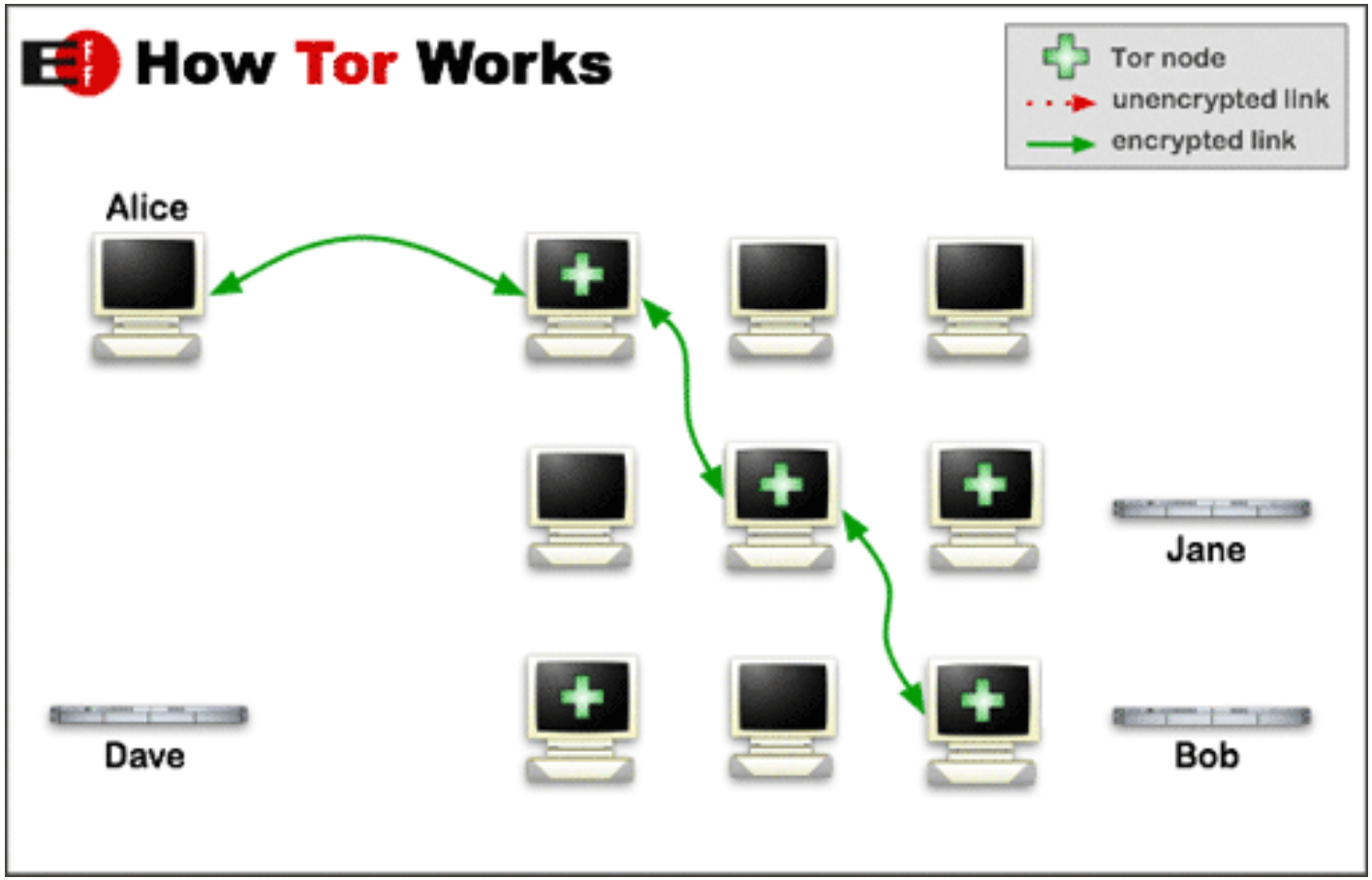
Exchange symmetric key with entry node (Diffie-Hellman)

Establishing a Circuit



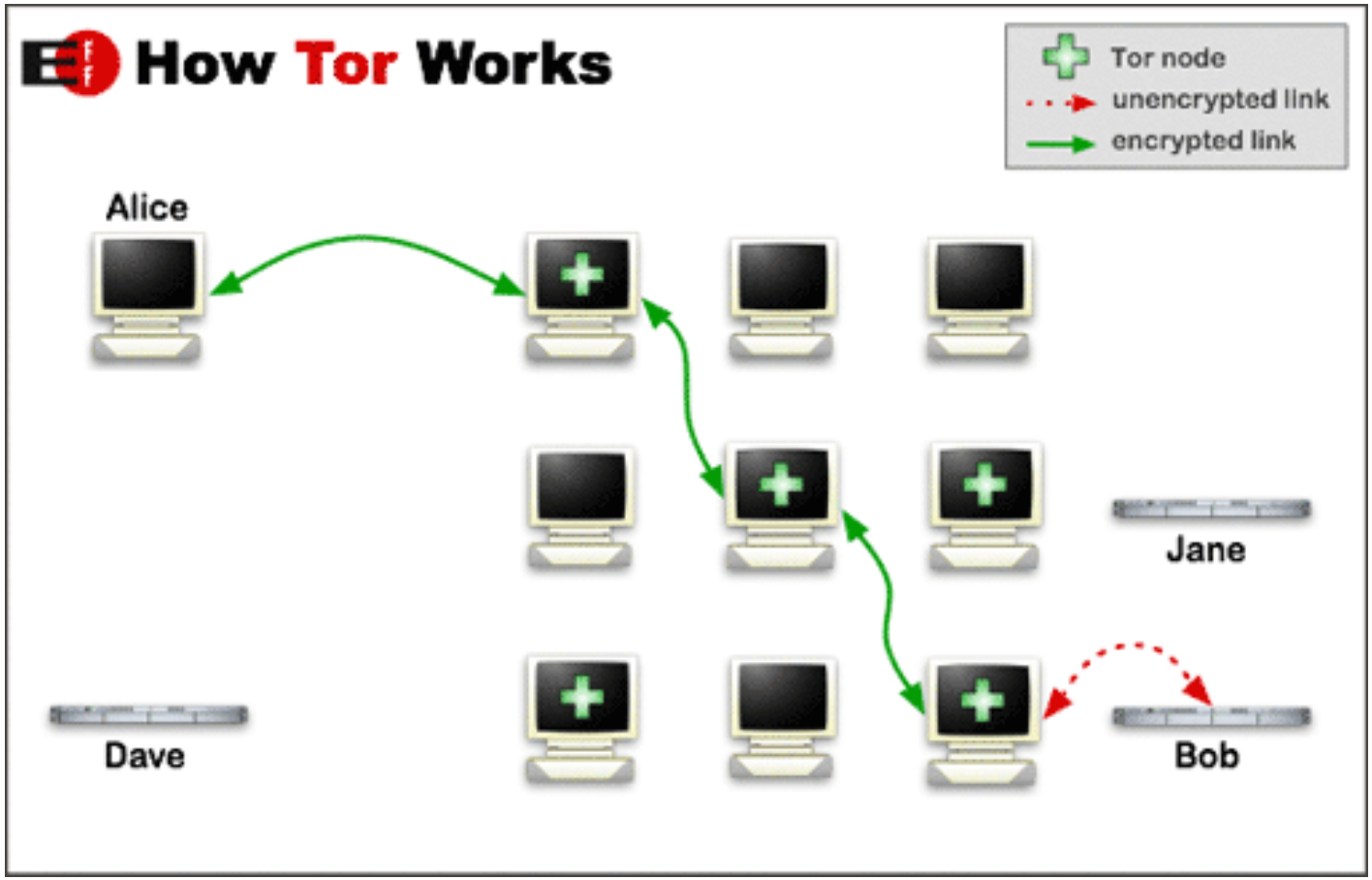
Exchange key with guard node (proxied by entry node!)

Establishing a Circuit



Exchange key with exit node (proxied by entry and guard node!)

Establishing a Circuit



Communicate with Bob (`www.wikileaks.org`)

Attacks against Tor, part I

- Tor offers anonymity up to the transport layer
- Tor cannot offer application-level anonymity
- Example: I connect through Tor to a website and enter on that website:

  "My name is Peter Schwabe, I live in the Netherlands, my IP address is 83.163.166.232."

Attacks against Tor, part I

- Tor offers anonymity up to the transport layer
- Tor cannot offer application-level anonymity
- Example: I connect through Tor to a website and enter on that website:

  "My name is Peter Schwabe, I live in the Netherlands, my IP address is 83.163.166.232."
- Various Bittorrent clients do precisely this: send the IP address as part of application data
- Conclusion: Bittorrent over Tor isn't a good idea

Attacks against Tor, part I

- Tor offers anonymity up to the transport layer
- Tor cannot offer application-level anonymity
- Example: I connect through Tor to a website and enter on that website:

  "My name is Peter Schwabe, I live in the Netherlands, my IP address is 83.163.166.232."
- Various Bittorrent clients do precisely this: send the IP address as part of application data
- Conclusion: Bittorrent over Tor isn't a good idea
- Browsers are easily identifiable, see Panopticlick by EFF
- Conclusion: Use the Tor browser (modified Firefox)

Attacks against Tor, part I

- Tor offers anonymity up to the transport layer
- Tor cannot offer application-level anonymity
- Example: I connect through Tor to a website and enter on that website:

  "My name is Peter Schwabe, I live in the Netherlands, my IP address is 83.163.166.232."
- Various Bittorrent clients do precisely this: send the IP address as part of application data
- Conclusion: Bittorrent over Tor isn't a good idea
- Browsers are easily identifiable, see Panopticlick by EFF
- Conclusion: Use the Tor browser (modified Firefox)
- Tor re-uses an existing circuit for new TCP connections for 10 minutes
- Leaking your IP address to Bittorrent may also de-anonymize your browser session (bad apple attack)!

Attacks against Tor, part II

- Tor provides anonymity as long as *not all three* relays attack *together*
- Possible attack: control all three relays on a path
- Anybody can run Tor relays, so can, for example, the NSA

Attacks against Tor, part II

- Tor provides anonymity as long as *not all three* relays attack *together*
- Possible attack: control all three relays on a path
- Anybody can run Tor relays, so can, for example, the NSA
- Let's assume that NSA runs 1% of the Tor relays
- Each circuit has a $1/1,000,000$ chance to be fully controlled by NSA

Attacks against Tor, part II

- Tor provides anonymity as long as *not all three* relays attack *together*
- Possible attack: control all three relays on a path
- Anybody can run Tor relays, so can, for example, the NSA
- Let's assume that NSA runs 1% of the Tor relays
- Each circuit has a $1/1,000,000$ chance to be fully controlled by NSA
- Possible solution: longer circuits (problem: slower, less reliable)

Attacks against Tor, part II

- Tor provides anonymity as long as *not all three* relays attack *together*
- Possible attack: control all three relays on a path
- Anybody can run Tor relays, so can, for example, the NSA
- Let's assume that NSA runs 1% of the Tor relays
- Each circuit has a $1/1,000,000$ chance to be fully controlled by NSA
- Possible solution: longer circuits (problem: slower, less reliable)
- Better solution: more non-NSA relays

Correlation attacks

- Tor is aiming at low latency (for web browsing etc.)
- Tor does not wait for traffic to do mixing

Correlation attacks

- Tor is aiming at low latency (for web browsing etc.)
- Tor does not wait for traffic to do mixing
- (Timing) correlation attack is still possible:
  - Think of the whole Tor network as one big proxy
  - Correlate traffic going into and out of this proxy

Correlation attacks

- Tor is aiming at low latency (for web browsing etc.)
- Tor does not wait for traffic to do mixing
- (Timing) correlation attack is still possible:
  - Think of the whole Tor network as one big proxy
  - Correlate traffic going into and out of this proxy
- Conclusion by Felix von Leitner (Fefe) on Aug 5, 2013:

  "Tor ist tot. Tor basiert auf der Annahme, dass der Gegner nicht in der Lage ist, das gesamte Internet zu überwachen."

Correlation attacks

- Tor is aiming at low latency (for web browsing etc.)
- Tor does not wait for traffic to do mixing
- (Timing) correlation attack is still possible:
  - Think of the whole Tor network as one big proxy
  - Correlate traffic going into and out of this proxy
- Conclusion by Felix von Leitner (Fefe) on Aug 5, 2013:

  "Tor is dead. Tor is based on the assumption, that the opponent does not have the whole Internet under surveillance"

Correlation attacks

- Tor is aiming at low latency (for web browsing etc.)
- Tor does not wait for traffic to do mixing
- (Timing) correlation attack is still possible:
  - Think of the whole Tor network as one big proxy
  - Correlate traffic going into and out of this proxy
- Conclusion by Felix von Leitner (Fefe) on Aug 5, 2013:

  "Tor is dead. Tor is based on the assumption, that the opponent does not have the whole Internet under surveillance"
- Very controversial discussion ensued... see
  http://blog.fefe.de/?ts=af0134f5

“Tor stinks”

- Snowden leaked NSA slides “Tor stinks” from 2007
- Quotes from these slides:

  “We will never be able to de-anonymize all Tor users all the time.”

  *“With manual analysis we can de-anonymize a **very small fraction** of Tor users, however **no** success de-anonymizing a user in response to a TOPI request/on demand.”*

Tor as censorship circumvention

- Various countries filter Internet traffic by destination address
- Most prominent example: Golden Shield Project ("Great Firewall of China")

Tor as censorship circumvention

- Various countries filter Internet traffic by destination address
- Most prominent example: Golden Shield Project ("Great Firewall of China")
- Firewalls and gateways cannot see the true destination of Tor traffic
- Tor is a powerful tool to circumvent censorship (e.g., in China)

Tor as censorship circumvention

- Various countries filter Internet traffic by destination address
- Most prominent example: Golden Shield Project ("Great Firewall of China")
- Firewalls and gateways cannot see the true destination of Tor traffic
- Tor is a powerful tool to circumvent censorship (e.g., in China)
- Can also use Tor to circumvent country filters:
  - Need an IP address in the US: use Tor with US exit node

Tor as censorship circumvention

- Various countries filter Internet traffic by destination address
- Most prominent example: Golden Shield Project ("Great Firewall of China")
- Firewalls and gateways cannot see the true destination of Tor traffic
- Tor is a powerful tool to circumvent censorship (e.g., in China)
- Can also use Tor to circumvent country filters:
  - Need an IP address in the US: use Tor with US exit node
  - Need access to a specific paper: use Tor with exit node in some university

Bridges and pluggable transports

- Easy solution for censors:
  - Obtain list of Tor nodes from directory server
  - Block access to the Tor network (all relays)

Bridges and pluggable transports

- Easy solution for censors:
  - Obtain list of Tor nodes from directory server
  - Block access to the Tor network (all relays)
- Solution: Tor bridges (entry notes that are not in the public list)
- Obtain IP address of a bridge by
  - visiting https://bridges.torproject.org/
  - writing e-mail to bridges@torproject.org

Bridges and pluggable transports

- Easy solution for censors:
  - Obtain list of Tor nodes from directory server
  - Block access to the Tor network (all relays)
- Solution: Tor bridges (entry notes that are not in the public list)
- Obtain IP address of a bridge by
  - visiting https://bridges.torproject.org/
  - writing e-mail to bridges@torproject.org
- Censors can also block Tor by identifying Tor traffic
- Tor traffic is relatively easy to identify:
  - Disguised as HTTPS traffic, but
  - uses random domain names
  - has a characteristic packet-size distribution

Bridges and pluggable transports

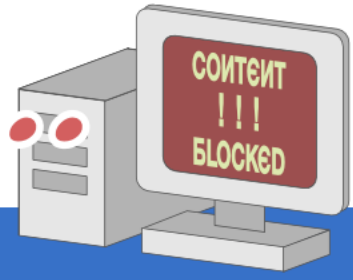
- Easy solution for censors:
  - Obtain list of Tor nodes from directory server
  - Block access to the Tor network (all relays)
- Solution: Tor bridges (entry notes that are not in the public list)
- Obtain IP address of a bridge by
  - visiting https://bridges.torproject.org/
  - writing e-mail to bridges@torproject.org
- Censors can also block Tor by identifying Tor traffic
- Tor traffic is relatively easy to identify:
  - Disguised as HTTPS traffic, but
  - uses random domain names
  - has a characteristic packet-size distribution
- Solution: fully disguise Tor traffic as other traffic
- Pluggable Transport API allows communication between ofuscator and Tor client

★ YOU ★

Can Help Protect

Freedom Of Speech

Online...



Run a

TOR RELAY

Today!

★ ★ TorProject.org ★ ★