15-744: Computer Networking

L-25 Privacy

Announcements

- This is the last lecture!
  - No lectures next week
  - Let me know if you want to meet about project
- Final on Wednesday 12/7
- Plan for project:
  - Replace poster by short presentations on the last day of classes
  - Project report due Monday 12/12
    - 6-8 pages

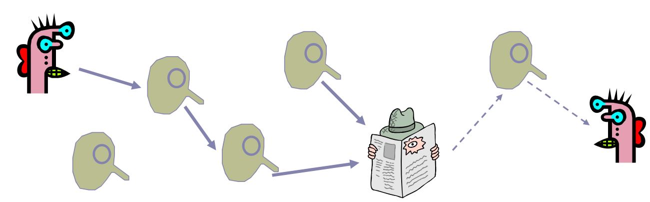
2

Overview

- Routing privacy
- Web Privacy
- Wireless Privacy

3

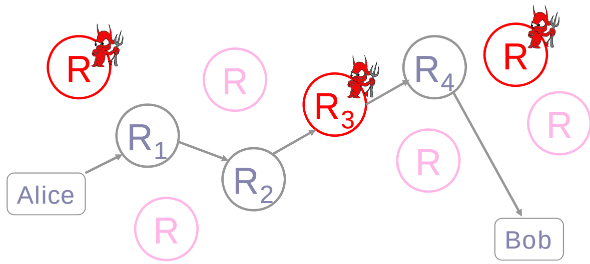
Randomized Routing



- Hide message source by routing it randomly
  - Popular technique: Crowds, Freenet, Onion routing
- Routers don’t know for sure if the apparent source of a message is the true sender or another router

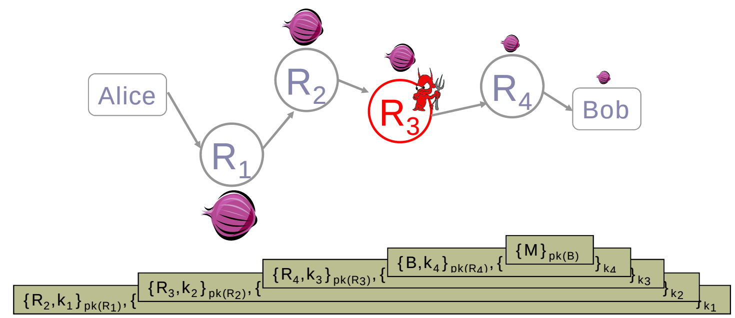
slide 4

Onion Routing



- Sender chooses a random sequence of routers
  - Some routers are honest, some controlled by attacker
  - Sender controls the length of the path

Route Establishment

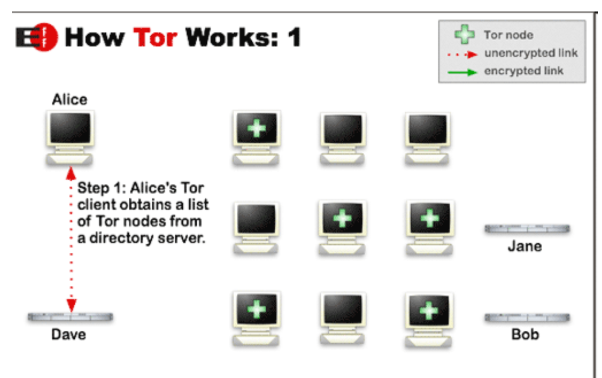


Routing info for each link encrypted with router's public key
Each router learns only the identity of the next router

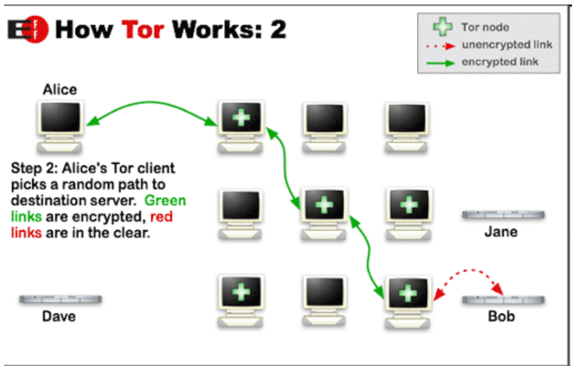
Tor

- Second-generation onion routing network
  - http://tor.eff.org
  - Developed by Roger Dingledine, Nick Mathewson and Paul Syverson
  - Specifically designed for low-latency anonymous Internet communications
- Running since October 2003
- 100s nodes on four continents, thousands of users
- "Easy-to-use" client proxy
  - Freely available, can use it for anonymous browsing

How does Tor work?

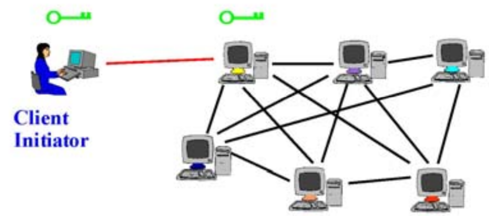


How does Tor work?



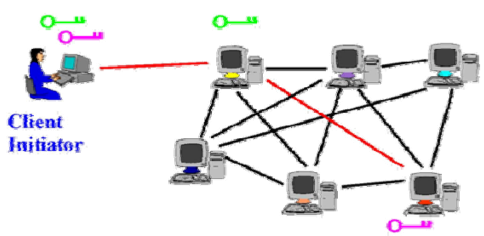
Tor Circuit Setup (1)

- Client proxy establish a symmetric session key and circuit with Onion Router #1



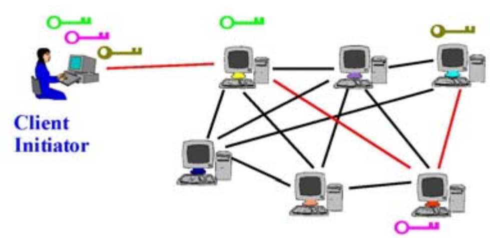
Tor Circuit Setup (2)

- Client proxy extends the circuit by establishing a symmetric session key with Onion Router #2
  - Tunnel through Onion Router #1 (don't need)



Tor Circuit Setup (3)

- Client proxy extends the circuit by establishing a symmetric session key with Onion Router #3
  - Tunnel through Onion Routers #1 and #2



Using a Tor Circuit

- Client applications connect and communicate over the established Tor circuit

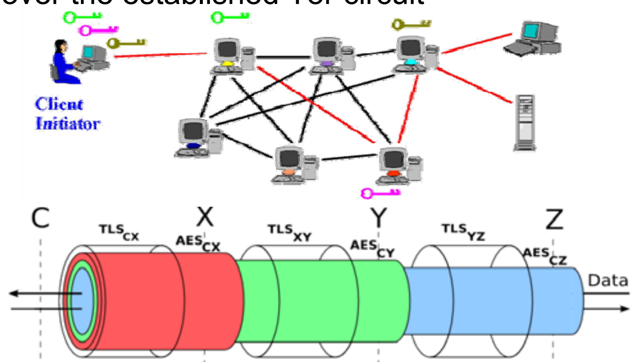
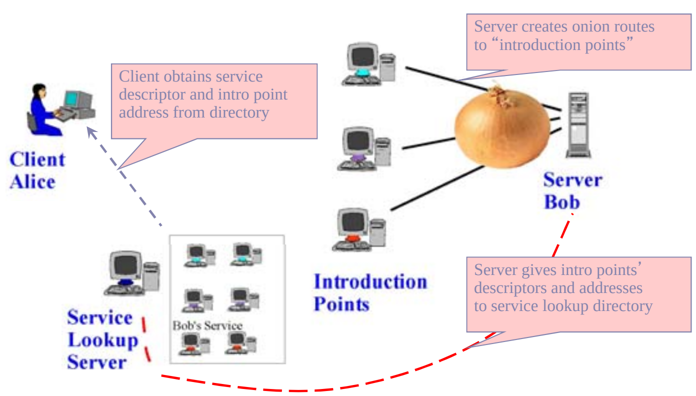


Image courtesy of Steven Murdoch

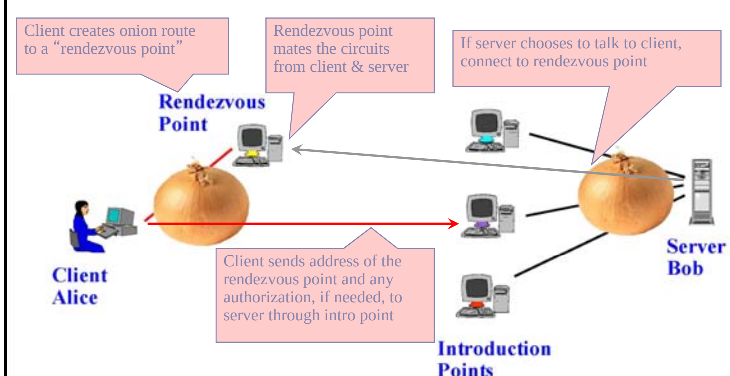
Location Hidden Servers

- Goal: deploy a server on the Internet that anyone can connect to without knowing where it is or who runs it
- Accessible from anywhere
- Resistant to censorship
- Can survive full-blown DoS attack
- Resistant to physical attack
  - Can't find the physical server!

Creating a Location Hidden Server



Using a Location Hidden Server



Overview

- Routing privacy
- Web Privacy
- Wireless Privacy

An "Old" Problem

- Many governments/companies trying to limit their citizens' access to information
  - Censorship (prevent access)
  - Punishment (deter access)
  - China, Saudi Arabia, HP
- How can we defeat such attempts?
  - Circumvent censorship
  - Undetectably

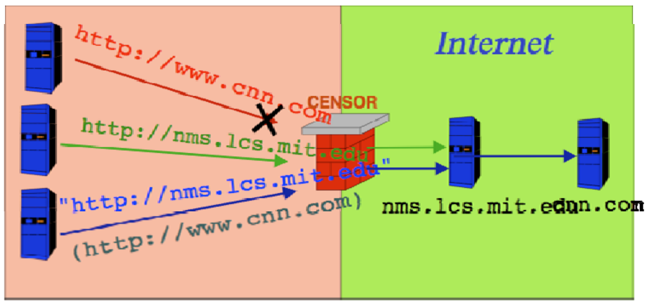
Proxy-Based Web Censorship

- Government manages national web firewall
  - Not optional---catches ALL web traffic
- Block certain requests
  - Possibly based on content
  - More commonly on IP address/publisher
  - China: Western news sites, Taiwan material
- Log requests to detect troublemakers
  - Even without blocking, may just watch traffic
- But they don't turn off the whole net
  - Creates a crack in their barrier

Goal

- Circumvent censor via innocent web activity
  - Normal web server and client cooperate to create covert channel
- Without consequence for client
- And without consequence for server
  - Broad participation increases system robustness
  - Ensure offering service doesn't lead to trouble
    - e.g., loss of business through being blocked
  - Also, "law knows no boundaries"

The Big Picture



21

Requirements

- Client deniability
  - Detection could be embarrassing or worse
- Client statistical deniability
  - Even suspicion could be a problem
- Server covertness/statistical deniability
  - If server detected, can be blocked
- Communication robustness
  - Even without detecting, censor could scramble covert channel
- Performance (bandwidth, latency)

22

(Un)related Work

- SSL
  - Encrypted connection---can't tell content
  - Suspicious!
  - Doesn't help reach blocked servers
  - Govt. can require revealing SSL keys
- Anonymizing Proxies
  - Prevent servers from knowing identity of client
  - But proxy inside censor can't reach content
  - And proxy outside censor can be blocked
  - And use of proxy is suspicious

23

Safeweb/Triangle boy

- Operation
  - Client contacts triangle-boy "reflector"
  - Reflector forwards requests to blocked server
  - Server returns content to client (IP spoof)
- Circumvents censorship
- But still easily detected
  - "Local monitoring of the user only reveals an encrypted conversation between User and Triangle Boy machine." (Safeweb manual)

24

Summary

- Easy to hide what you are getting
  - Just use SSL
- And easy to circumvent censors
  - Safeweb
- But hard to hide that you are doing it

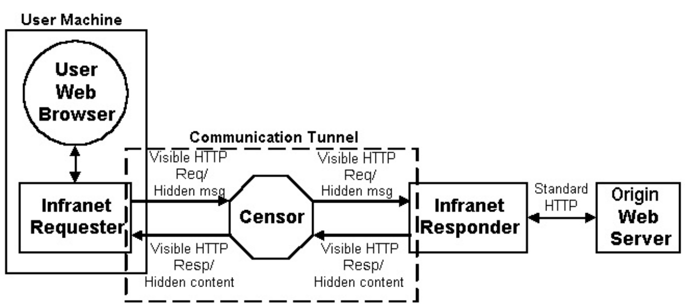
25

Infranet: Circumventing Censors

- Censors allow certain traffic
- Use to construct a covert channel
  - Talk to normal servers
  - Embed requests for censored content in normal-seeming requests
  - Receive censored content hidden in normal-seeming responses
- Requester: client asking for hidden content
- Responder: server covertly providing it

26

System Architecture



27

Receiving Content is Easier Half

- Responder is a normal web server, serving images (among other things)
- Encrypt data using requestor key
- Embed in "unimportant, random" bits of images
  - E.g., high order color bits
  - Watermarking
- Encrypted data looks random---only requestor can tell it isn't (and decrypt)

28

Example


- One image has embedded content
- You can't tell which (shows it's working)

Goals Analysis

- Client looks innocent (receives images)
  - Infranet users & nonusers indistinguishable
- Server less so
  - Any one image seems innocent
  - But same image with different "random bits" in each copy is suspicious
  - Evasion: never use same image-URL twice
    - Justify: per-individual customized web site
    - Human inspection might detect odd URL usage
  - Evasion: use time-varying image (webcam)
- Performance: 1/3 of image bits

Upstream (Requests) is Harder

- No "random content bits" that can be fiddled to send messages to responder
- Solution: let browsing pattern itself be the message
- Suppose web page has k links.
  - GET on i^{th} link signifies symbol "i" to requestor
  - Result: $\log_2(k)$ message bits from link click
- Can be automated
- To prevent censor from seeing message, encrypt with responder key

Goals Analysis

- Deniability: requestor generates standard http GETs to allowed web sites
  - Fact of GETs isn't itself proof of wrongdoing
  - Known rule for translating GETs to message, but message is encrypted, so not evidence
- Statistical deniability
  - Encrypting message produces "random" string
  - Sent via series of "random" GETs
  - Problem: normal user browsing not random
    - Some links rare
    - Conditional dependence of browsing on past browsing

Performance vs. Deniability

- Middling deniability, poor performance
  - Request URL may be (say) 50 characters
  - 16 Links/Page (say) means 4 bits
  - So need 100 GETs to request one URL!
  - And still poor statistical deniability
- Can we enhance deniability?
  - Yes, by decreasing performance further
- Can we enhance performance?
  - Yes, and enhance deniability at same time

33

Paranoid Alternative

- Settle for one message bit per GET
  - Odd/even links on page
  - Or generalize to "mod k" for some small k
- User has many link choices for each bit
  - Can choose one that is reasonable
  - Incorporate error correcting code in case no reasonable next link sends correct bit
- Drawback: user must be directly involved in sending each message bit
  - Very low bandwidth vs time spent

34

Higher Performance

- Idea: arithmetic coding of requests
  - If request i has probability p_i, then entropy of request distribution is $-\Sigma\ p_i \log p_i$
  - Arithmetic coding encodes request i using $\log p_i$ bits
  - Result: expected request size equals entropy
  - Optimal
- Problem: requestor does not know the probability distribution of requests
  - Does not have info needed for encoding

35

Solution: Range Mapping

- Adler-Maggs
- Exploit asymmetric bandwidth
- Responder sends probability distribution to requester using easy, downstream path
- Requestor uses this "dictionary" to build arithmetic code, send encoded result
- Variation for non-binary
  - Our messages aren't bits, they are clicks
  - And server knows different clicks should have different probabilities

36

Toy Example

- Suppose possible requests fewer than links on page
- Responder sends dictionary:
  - “link 1 means http://mit.edu”
  - “link 2 means http://stanford.edu”
  - Assigns common requests to common GETs
- Requestor GETs link matching intended request
- One GET sends full (possibly huge) request
- Problem: in general, ∞ possible requests
  - Can’t send a dictionary for all

37

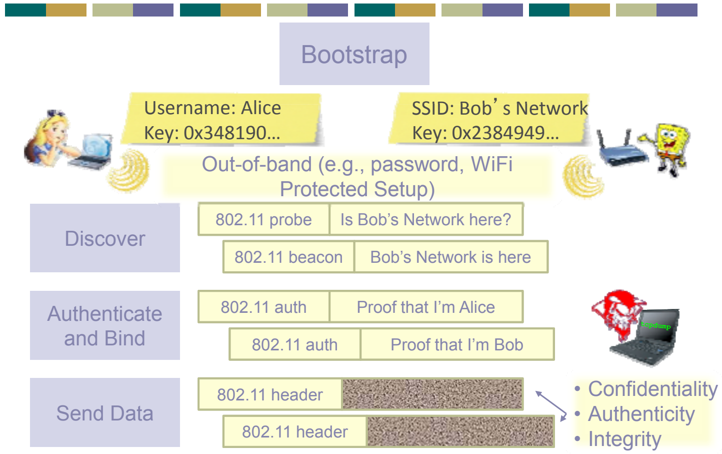
Overview

- Routing privacy
- Web Privacy
- Wireless Privacy
  - Slides Jamie

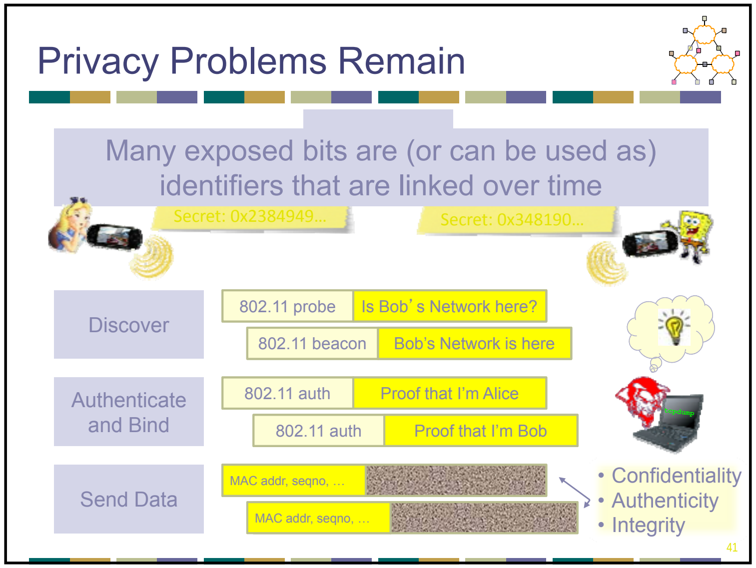
38

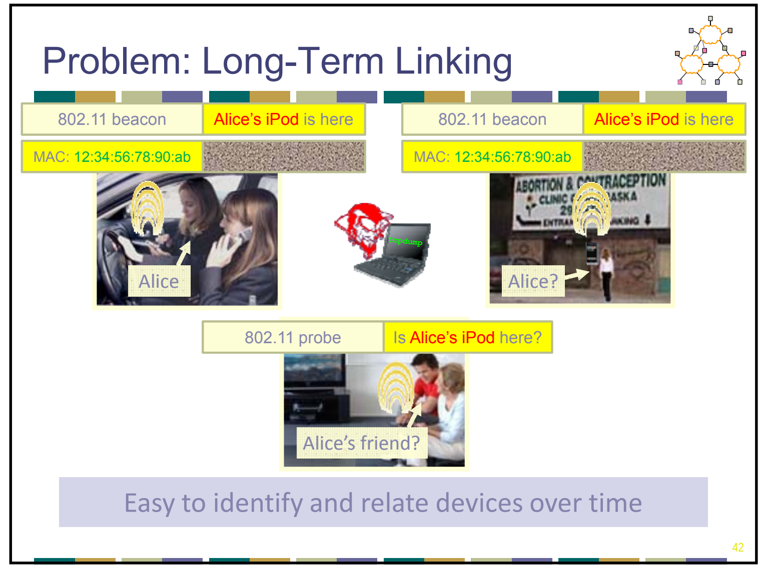
Our Wireless World

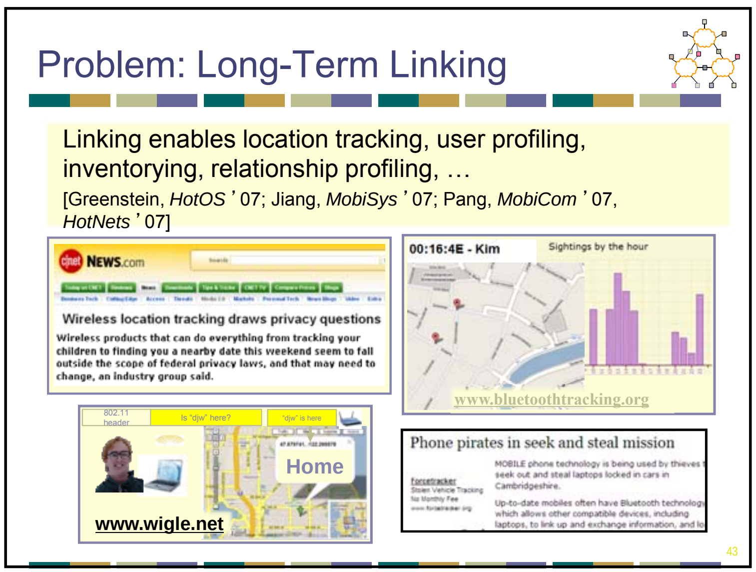
Best Security Practices

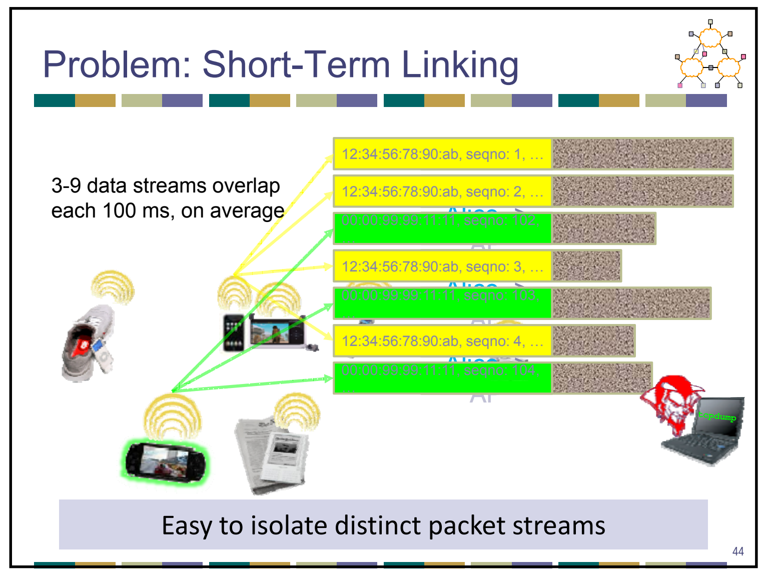


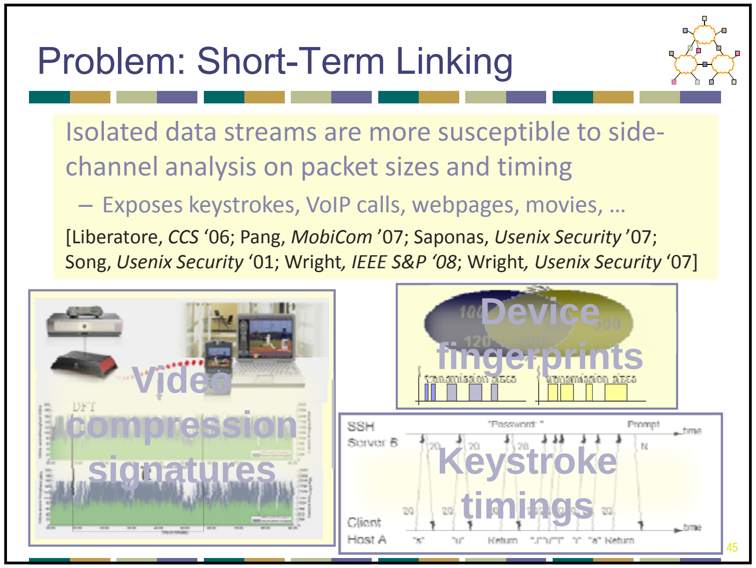
40

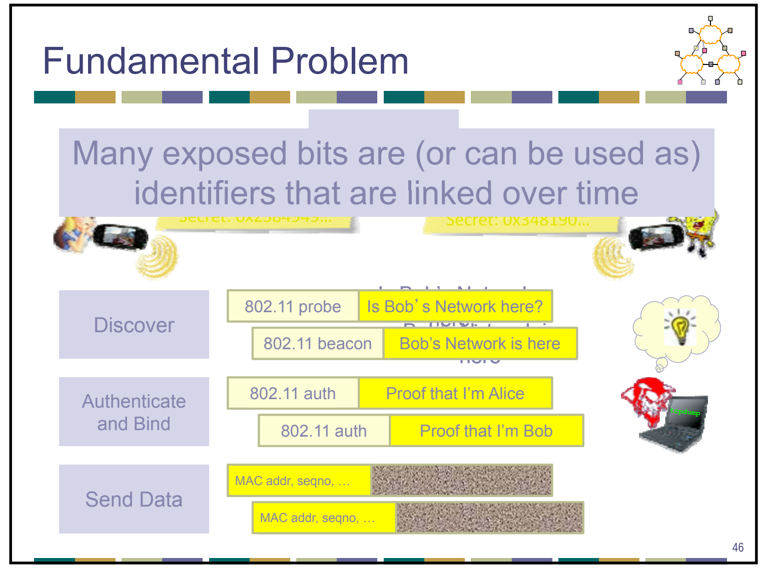

Privacy Problems Remain
Many exposed bits are (or can be used as) identifiers that are linked over time
Secret: 0x2384949…
Secret: 0x348190…
Discover
802.11 probe
Is Bob's Network here?
802.11 beacon
Bob's Network is here
Authenticate and Bind
802.11 auth
Proof that I'm Alice
802.11 auth
Proof that I'm Bob
Send Data
MAC addr, seqno, …
MAC addr, seqno, …
• Confidentiality
• Authenticity
• Integrity
41

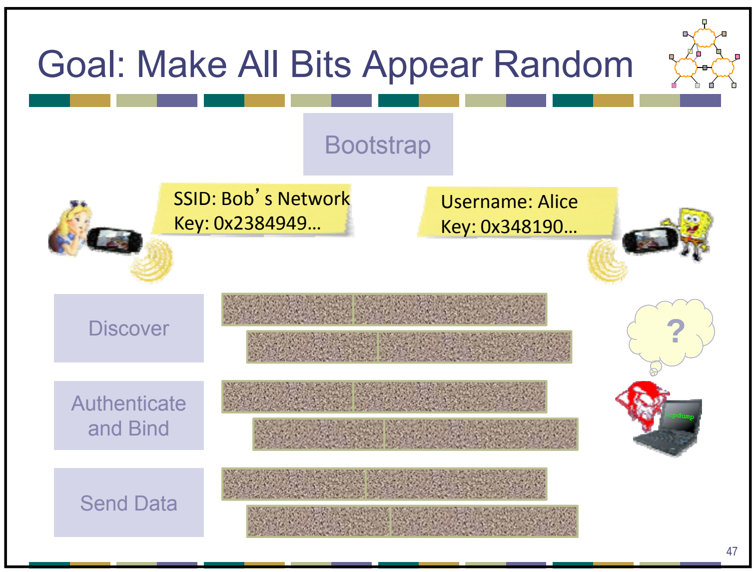

Problem: Long-Term Linking
802.11 beacon
Alice's iPod is here
MAC: 12:34:56:78:90:ab
802.11 beacon
Alice's iPod is here
MAC: 12:34:56:78:90:ab
Alice
Alice?
ABORTION & CONTRACEPTION
802.11 probe
Is Alice's iPod here?
Alice's friend?
Easy to identify and relate devices over time
42

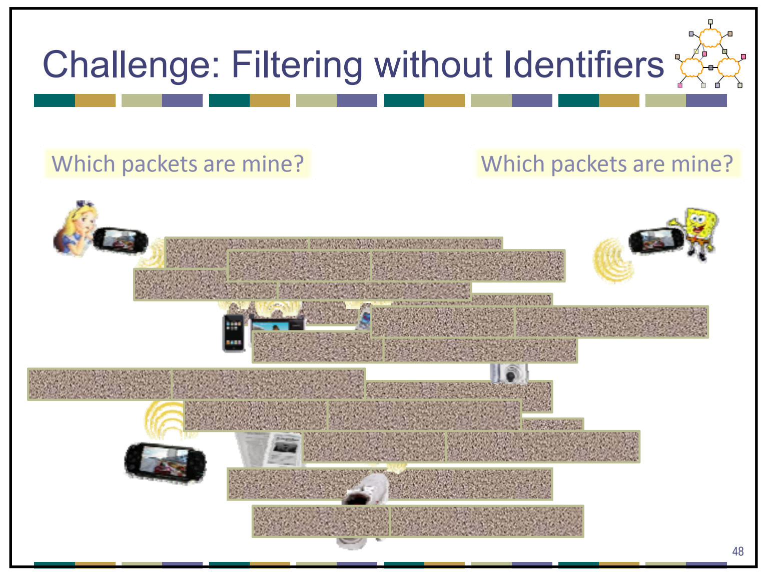

Problem: Long-Term Linking
Linking enables location tracking, user profiling, inventorying, relationship profiling, …
[Greenstein, HotOS '07; Jiang, MobiSys '07; Pang, MobiCom '07, HotNets '07]
NEWS.com
Wireless location tracking draws privacy questions
Wireless products that can do everything from tracking your children to finding you a nearby date this weekend seem to fall outside the scope of federal privacy laws, and that may need to change, an industry group said.
00:16:4E - Kim
Sightings by the hour
www.bluetoothtracking.org
Is "djw" here?
"djw" is here
Home
www.wigle.net
Phone pirates in seek and steal mission
43


Problem: Short-Term Linking
3-9 data streams overlap each 100 ms, on average
12:34:56:78:90:ab, seqno: 1, …
12:34:56:78:90:ab, seqno: 2, …
12:34:56:78:90:ab, seqno: 3, …
12:34:56:78:90:ab, seqno: 4, …
Easy to isolate distinct packet streams
44


Problem: Short-Term Linking
Isolated data streams are more susceptible to side-channel analysis on packet sizes and timing
– Exposes keystrokes, VoIP calls, webpages, movies, …
[Liberatore, CCS '06; Pang, MobiCom '07; Saponas, Usenix Security '07; Song, Usenix Security '01; Wright, IEEE S&P '08; Wright, Usenix Security '07]
Video compression signatures
Device fingerprints
Keystroke timings
SSH Server S
Client Host A
45


Fundamental Problem
Many exposed bits are (or can be used as) identifiers that are linked over time
Discover
Authenticate and Bind
Send Data
802.11 probe
Is Bob's Network here?
802.11 beacon
Bob's Network is here
802.11 auth
Proof that I'm Alice
802.11 auth
Proof that I'm Bob
MAC addr, seqno, …
MAC addr, seqno, …
46


Goal: Make All Bits Appear Random
Bootstrap
SSID: Bob's Network
Key: 0x2384949…
Username: Alice
Key: 0x348190…
Discover
Authenticate and Bind
Send Data
?
47


Challenge: Filtering without Identifiers
Which packets are mine?
Which packets are mine?
48

Design Requirements

- When *A* generates *Message* to *B*, she sends:

PrivateMessage = **F**(*A*, *B*, *Message*)

A→B | Header… | Unencrypted payload

where **F** has these properties:

- **Confidentiality**: Only *A* and *B* can determine *Message*.
- **Authenticity**: *B* can verify *A* created *PrivateMessage*.
- **Integrity**: *B* can verify *Message* not modified.
- **Unlinkability**: Only *A* and *B* can link *PrivateMessages* to same sender or receiver.
- **Efficiency**: *B* can process *PrivateMessages* as fast as he can receive them.

49

Solution Summary

	Confidentiality	Authenticity	Integrity	Unlinkability	Efficiency
802.11 WPA	Only Data Payload	Only Data Payload	Only Data Payload	🚫	✓
MAC Pseudonyms					
Public Key Symmetric Key					
SlyFi: Discovery/Binding					
SlyFi: Data packets					

50

Straw man: MAC Pseudonyms

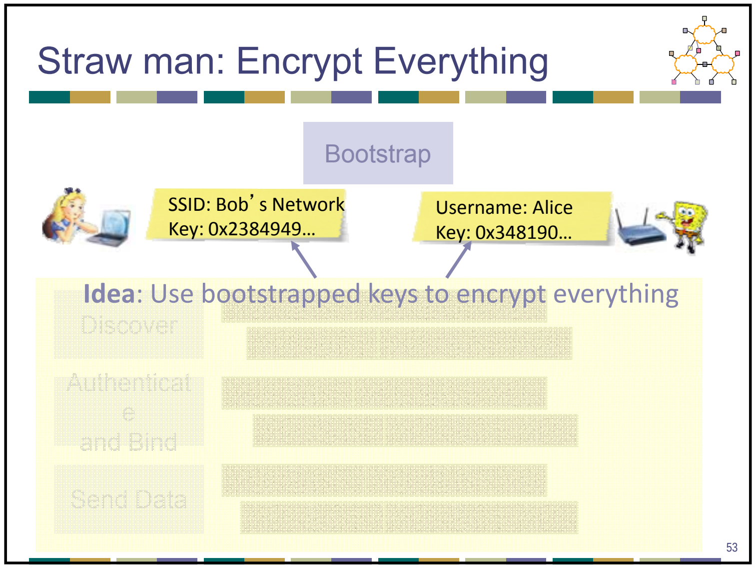
- **Idea**: change MAC address periodically
  - Per session or when idle [Gruteser'05, Jiang '07]
- Other fields remain (e.g., in discovery/binding)
  - No mechanism for data authentication/encryption
  - Doesn't hide network names during discovery or credentials during authentication
- Pseudonyms are linkable in the short-term
  - Same MAC must be used for each association
  - Data streams still vulnerable to side-channel leaks

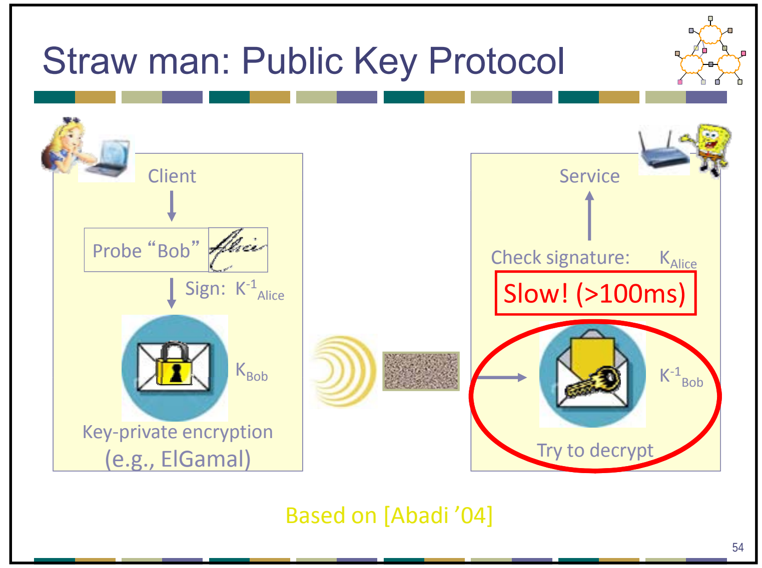
51

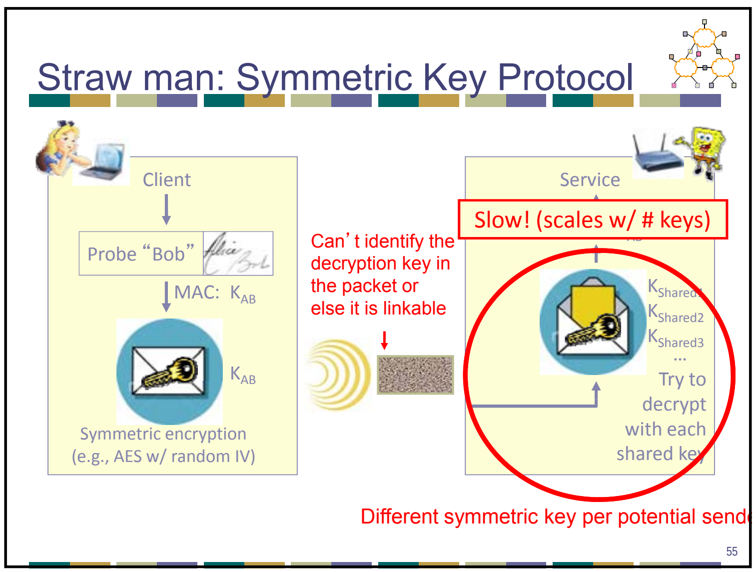
Solution Summary

	Confidentiality	Authenticity	Integrity	Unlinkability	Efficiency
802.11 WPA	Only Data Payload	Only Data Payload	Only Data Payload	🚫	✓
MAC Pseudonyms	🚫	🚫	🚫	Long Term	✓
Public Key Symmetric Key					
SlyFi: Discovery/Binding					
SlyFi: Data packets					

52







Solution Summary

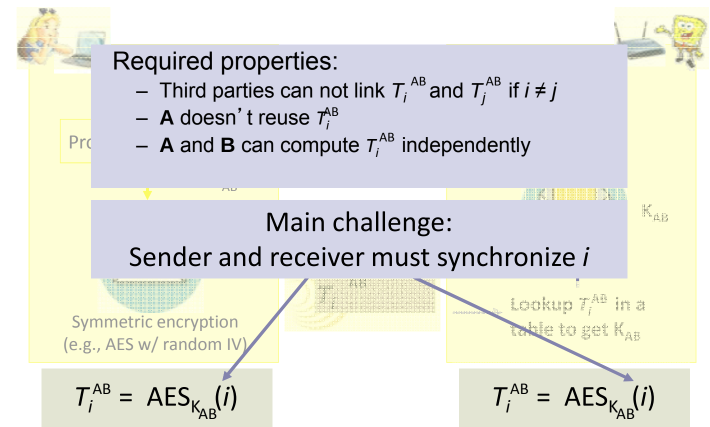
	Confidentiality	Authenticity	Integrity	Unlinkability	Efficiency
802.11 WPA	**Only Data Payload**	**Only Data Payload**	**Only Data Payload**	⊘	✓
MAC Pseudonyms	⊘	⊘	⊘	**Long Term**	✓
Public Key Protocol Symmetric Key Protocol	✓	✓	✓	✓	⊘
SlyFi: Discovery/Binding					
SlyFi: Data packets					

56

SlyFi

- Symmetric key almost works, but tension between:
  - Unlinkability: can't expose the identity of the key
  - Efficiency: need to identify the key to avoid trying all keys
- **Idea**: Identify the key in an unlinkable way
- Approach:
  - Sender **A** and receiver **B** agree on tokens: $T_1^{AB}, T_2^{AB}, T_3^{AB}, \ldots$
  - **A** attaches T_i^{AB} to encrypted packet for **B**

SlyFi



Required properties:
- Third parties can not link T_i^{AB} and T_j^{AB} if $i \neq j$
- **A** doesn't reuse T_i^{AB}
- **A** and **B** can compute T_i^{AB} independently

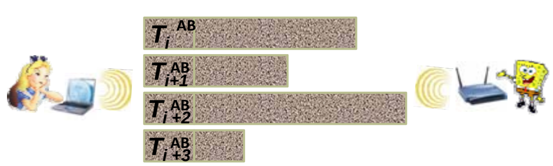
Main challenge:
Sender and receiver must synchronize *i*

$T_i^{AB} = AES_{K_{AB}}(i)$ $T_i^{AB} = AES_{K_{AB}}(i)$

SlyFi: Data Transport

- Data messages:
  - Only sent over established connections
  - ⇒ Expect messages to be delivered
  - ⇒ Use implicit transmission number to synchronize *i*

$T_i^{AB} = AES_{K_{AB}}(i)$ where i = transmission #



SlyFi: Data Transport

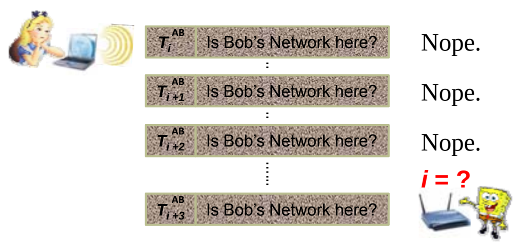
- Data messages:
  - Only sent over established connections
  - ⇒ Expect messages to be delivered
  - ⇒ Use implicit transmission number to synchronize *i*

$T_i^{AB} = AES_{K_{AB}}(i)$ where i = transmission #

- On receipt of T_i^{AB}, **B** computes next expected: T_{i+1}^{AB}
- Handling message loss:
  - On receipt of T_i^{AB} save $T_{i+1}^{AB}, \ldots, T_{i+k}^{AB}$ in table
  - Tolerates *k* consecutive losses (*k*=50 is enough)
  - No loss ⇒ compute one token per reception

SlyFi: Discovery/Binding

- Discovery & binding messages:
  - Often sent when other party is not present
  - ⇒ Can't expect most messages to be delivered
  - ⇒ Can't rely on transmission reception to synchronize *i*

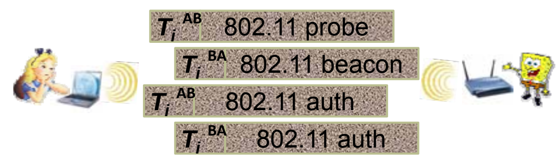


61

SlyFi: Discovery/Binding

- Discovery & binding messages:
  - **Infrequent**: only sent when trying to associate
  - **Narrow interface**: single application, few side-channels
  - ⇒ Linkability at short timescales is usually OK
  - ⇒ Use loosely synchronized time to synchronize *i*

T_i^{AB} = $AES_{K_{AB}}(i)$ where $i = \lfloor \text{current time}/5 \text{ min} \rfloor$



62

SlyFi: Discovery/Binding

- Discovery & binding messages:
  - **Infrequent**: only sent when trying to associate
  - **Narrow interface**: single application, few side-channels
  - ⇒ Linkability at short timescales is usually OK
  - ⇒ Use loosely synchronized time to synchronize *i*

T_i^{AB} = $AES_{K_{AB}}(i)$ where $i = \lfloor \text{current time}/5 \text{ min} \rfloor$

- At the start of time interval *i* compute T_i^{AB}
- Handling clock skew:
  - Receiver **B** saves $T_{i-s}^{AB}, \ldots, T_{i+s}^{AB}$ in table
  - Tolerates clock skew of $5 \cdot s$ minutes

63

Solution Summary

	Confidentiality	Authenticity	Integrity	Unlinkability	Efficiency
802.11 WPA	Only Data Payload	Only Data Payload	Only Data Payload	⃠	✓
MAC Pseudonyms	⃠	⃠	⃠	Long Term	✓
Public Key Symmetric Key	✓	✓	✓	✓	⃠
SlyFi: Discovery/Binding	✓	✓	✓	Long Term	✓
SlyFi: Data packets	✓	✓	✓	✓	✓

64

Next Lecture…

- No next lecture ☺
- Exam
  - Much like the midterm
  - 1.5hrs on Dec 1st
  - Mostly on 2nd half of semester but with some coverage of 1st half
- Project
  - 6-8pg writeup – due Dec 5th
  - 10min presentation – on Dec 2nd or 3rd
    - Incorporate feedback into final writeup

Overview

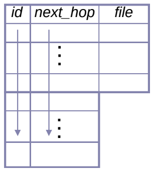
- P2P Privacy

Freenet

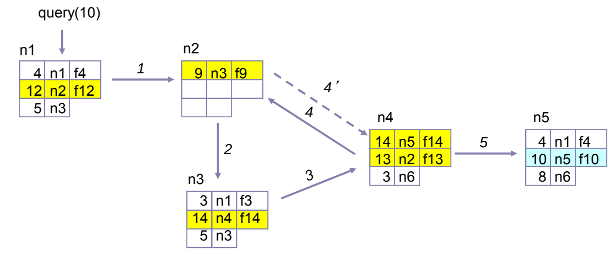
- Addition goals to file location:
  - Provide publisher anonymity, security
  - Resistant to attacks – a third party shouldn't be able to deny the access to a particular file (data item, object), even if it compromises a large fraction of machines
- Files are stored according to associated key
  - Core idea: try to cluster information about similar keys
- Messages
  - Random 64bit ID used for loop detection
  - TTL
    - TTL 1 are forwarded with finite probablity
    - Helps anonymity
  - Depth counter
    - Opposite of TTL – incremented with each hop
    - Depth counter initialized to small random value

Data Structure

- Each node maintains a common stack
  - *id* – file identifier
  - *next_hop* – another node that store the file id
  - *file* – file identified by *id* being stored on the local node
- Forwarding:
  - Each message contains the file *id* it is referring to
  - If file *id* stored locally, then stop
    - Forwards data back to upstream requestor
    - Requestor adds file to cache, adds entry in routing table
  - If not, search for the "closest" *id* in the stack, and forward the message to the corresponding *next_hop*



Query Example

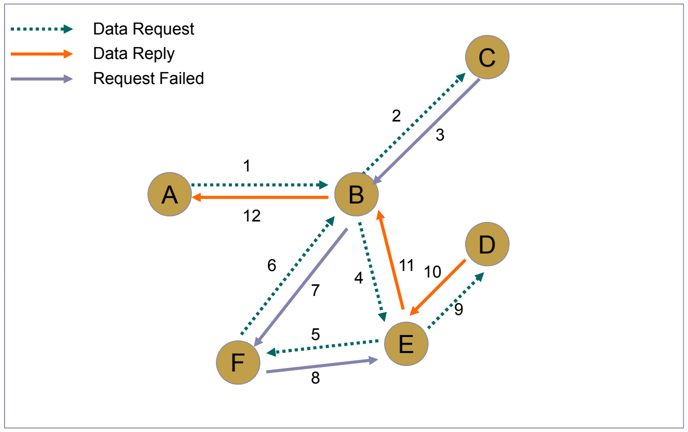


Note: doesn't show file caching on the reverse path

Freenet Requests

- Any node forwarding reply may change the source of the reply (to itself or any other node)
  - Helps anonymity
- Each query is associated a TTL that is decremented each time the query message is forwarded; to obscure distance to originator:
  - TTL can be initiated to a random value within some bounds
  - When TTL=1, the query is forwarded with a finite probability
- Each node maintains the state for all outstanding queries that have traversed it → help to avoid cycles
- If data is not found, failure is reported back
  - Requestor then tries next closest match in routing table

Freenet Request



Freenet Search Features

- Nodes tend to specialize in searching for similar keys over time
  - Gets queries from other nodes for similar keys
- Nodes store similar keys over time
  - Caching of files as a result of successful queries
- Similarity of keys does not reflect similarity of files
- Routing does not reflect network topology

73

Freenet File Creation

- Key for file generated and searched → helps identify collision
  - Not found ("All clear") result indicates success
  - Source of insert message can be change by any forwarding node
- Creation mechanism adds files/info to locations with similar keys
- New nodes are discovered through file creation
- Erroneous/malicious inserts propagate original file further

74

Cache Management

- LRU Cache of files
- Files are not guaranteed to live forever
  - Files "fade away" as fewer requests are made for them
- File contents can be encrypted with original text names as key
  - Cache owners do not know either original name or contents → cannot be held responsible

75

Freenet Naming

- Freenet deals with keys
  - But humans need names
  - Keys are flat → would like structure as well
- Could have files that store keys for other files
  - File /text/philiosophy could store keys for files in that directory → how to update this file though?
- Search engine → undesirable centralized solution

76

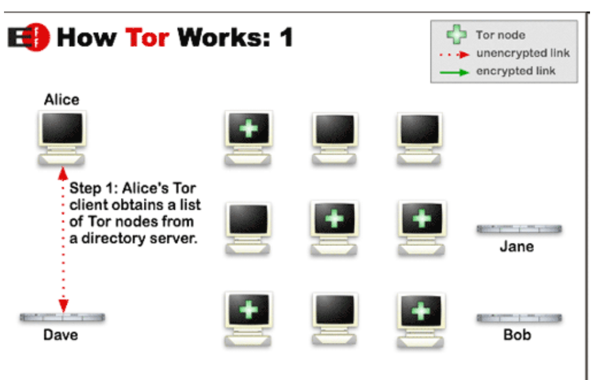
Freenet Naming - Indirect files

- Normal files stored using content-hash key
  - Prevents tampering, enables versioning, etc.
- Indirect files stored using name-based key
  - Indirect files store keys for normal files
  - Inserted at same time as normal file
- Has same update problems as directory files
  - Updates handled by signing indirect file with public/private key
  - Collisions for insert of new indirect file handled specially → check to ensure same key used for signing
- Allows for files to be split into multiple smaller parts

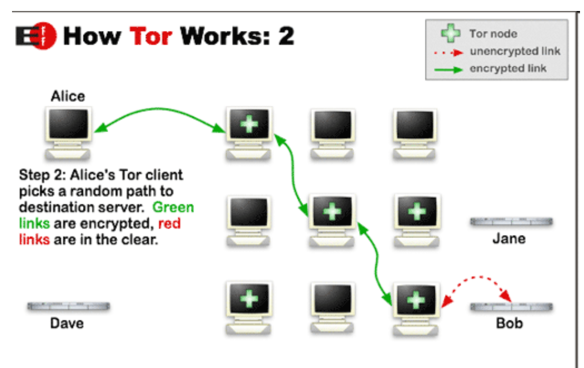
77

78

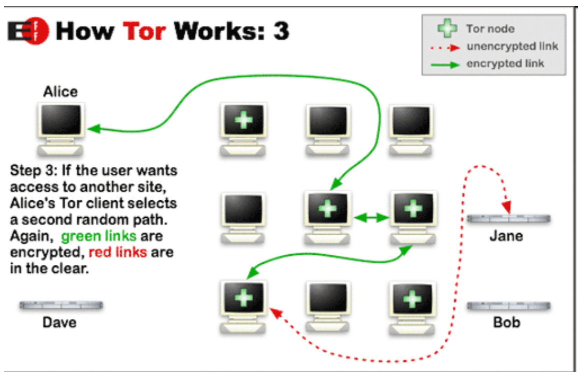
How does Tor work?



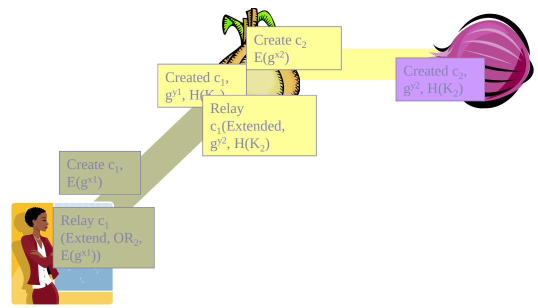
How does Tor work?



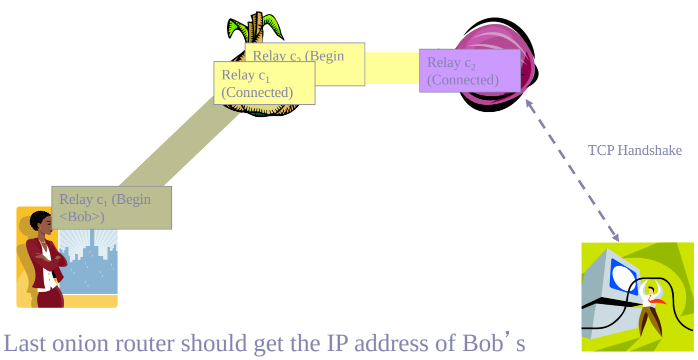
How does Tor work?



Building a circuit



Fetching a web page



Last onion router should get the IP address of Bob's website to protect Alice's anonymity.