15-440 Distributed Systems

24 – Security Protocols - II

Thursday, Nov 29th, 2018

Logistical Updates

- P3 FINAL – Due 12/1 (Saturday)
  - Please make sure your group information is correct!

- HW4 - Due 12/4 (Tuesday) **NO LATE DAYS**

- Midterm II – Review session, in class 12/4
  - Focus on topics on 2nd half of the class
  - Similar to the mid-term review.

- Midterm II – Next Thursday
  - **Location: CUC McConomy**, **Time:** 10:30am – 11:50am
  - If you need extra time, please send Instructors email
  - Please come 10mins early to get seated

Today's Lecture

- Effective secure channels
- Access control
- Privacy and Tor

The Great Divide

	Symmetric Crypto: (Private key) Example: AES	Asymmetric Crypto: (Public key) Example: RSA
Requires a pre-shared secret between communicating parties?	Yes	No
Overall speed of cryptographic operations	Fast	Slow

One last “little detail”…

How do I get these keys in the first place??

Remember:

- Symmetric key primitives assumed Alice and Bob had already shared a key.
- Asymmetric key primitives assumed Alice knew Bob’s public key.

This may work with friends, but when was the last time you saw Amazon.com walking down the street?

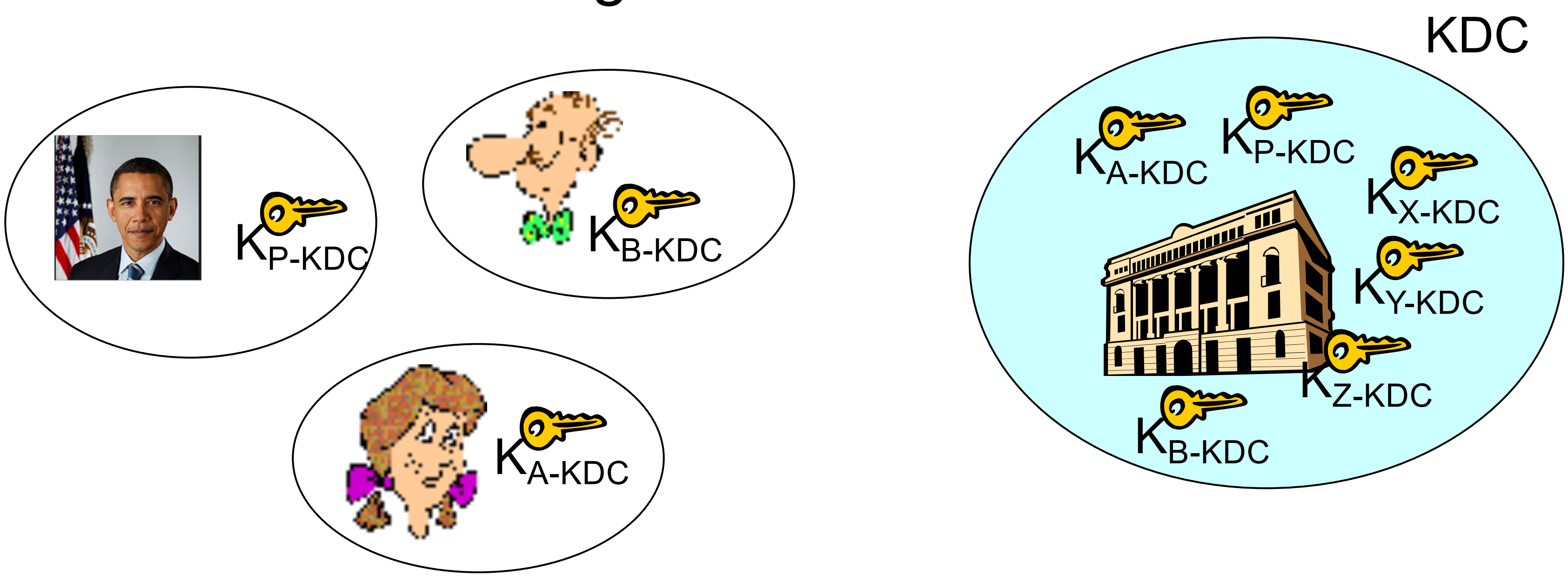
Recap: Symmetric Key Distribution

- How does Andrew do this?

Andrew Uses Kerberos, which relies on a Key Distribution Center (KDC) to establish shared symmetric keys.

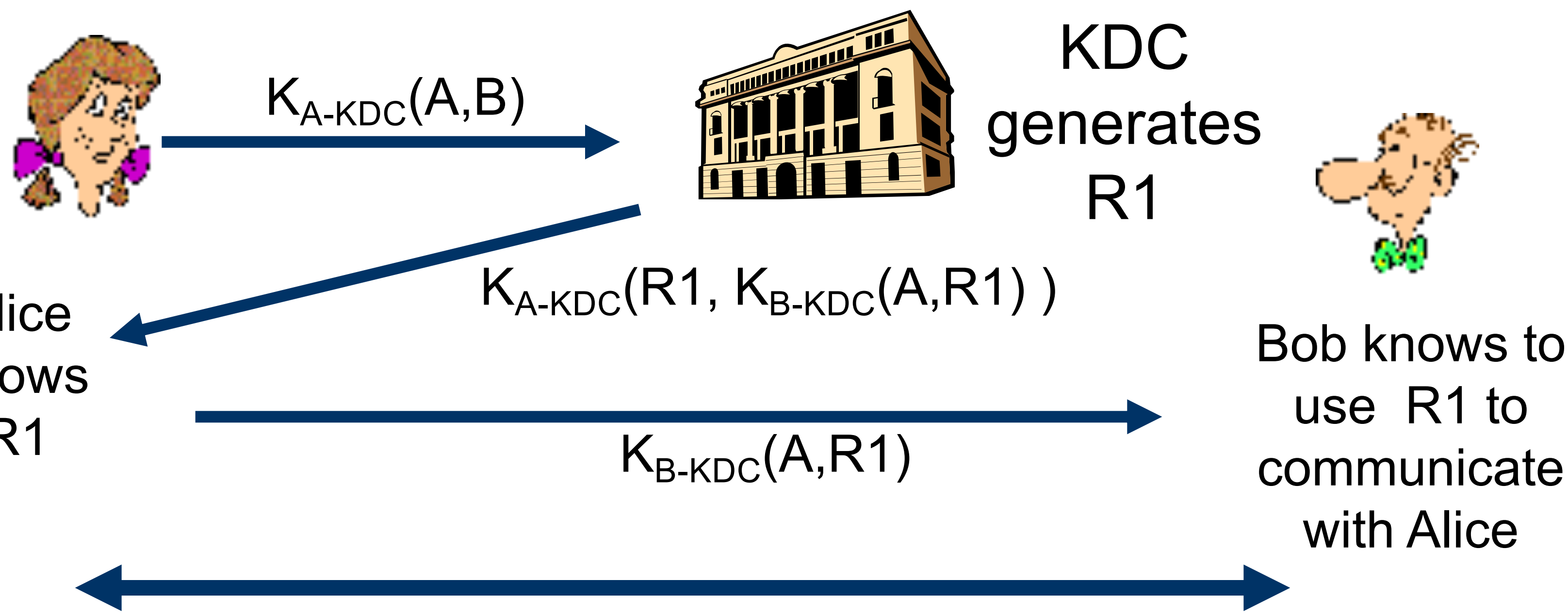
Key Distribution Center (KDC)

- Alice, Bob need shared <u>symmetric key</u>.
- KDC: server shares different secret key with *each* registered user (many users)
- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC.



Key Distribution Center (KDC)

Q: How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?



Alice and Bob communicate: using R1 as *session key* for shared symmetric encryption

How Useful is a KDC?

- Must always be online to support secure communication
- KDC can expose our session keys to others!
- Centralized trust and point of failure.

In practice, the KDC model is mostly used within single organizations (e.g. Kerberos) but not more widely.

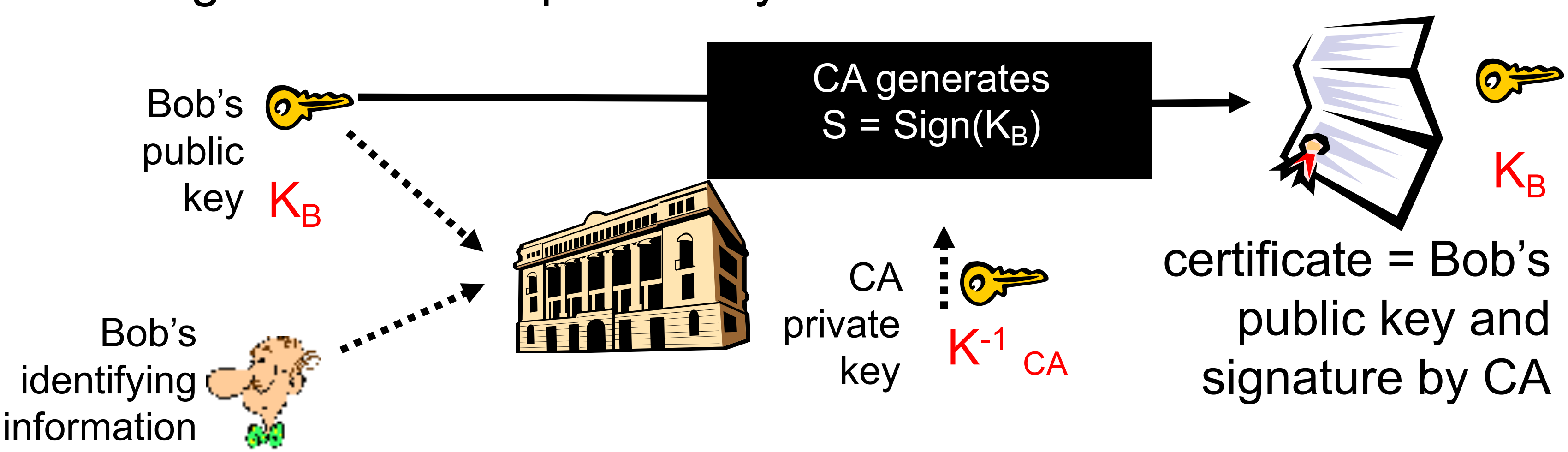
The Dreaded PKI

- Definition: Public Key Infrastructure (PKI)

1) A system in which "roots of trust" authoritatively bind public keys to real-world identities
2) A significant stumbling block in deploying many "next generation" secure Internet protocol or applications.

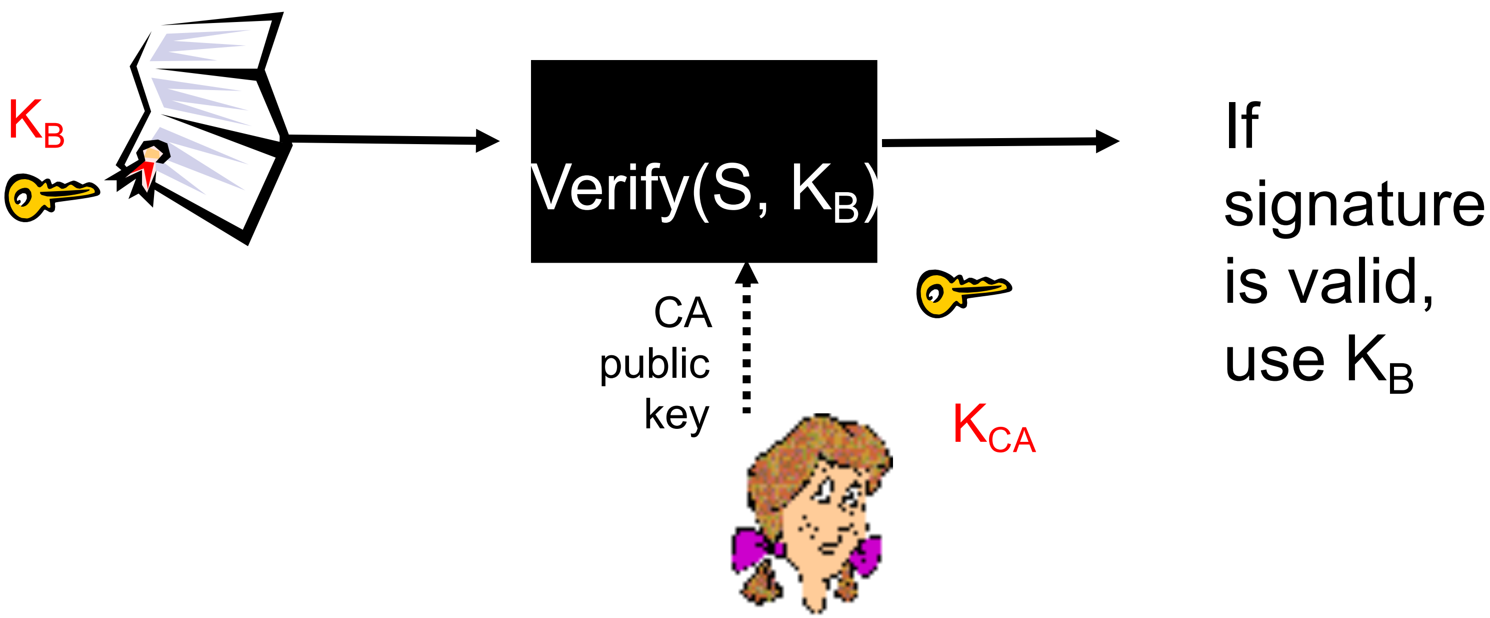
Certification Authorities

- Certification authority (CA): binds public key to particular entity, E.
- An entity E registers its public key with CA.
  - E provides “proof of identity” to CA.
  - CA creates certificate binding E to its public key.
  - Certificate contains E’s public key AND the CA’s signature of E’s public key.



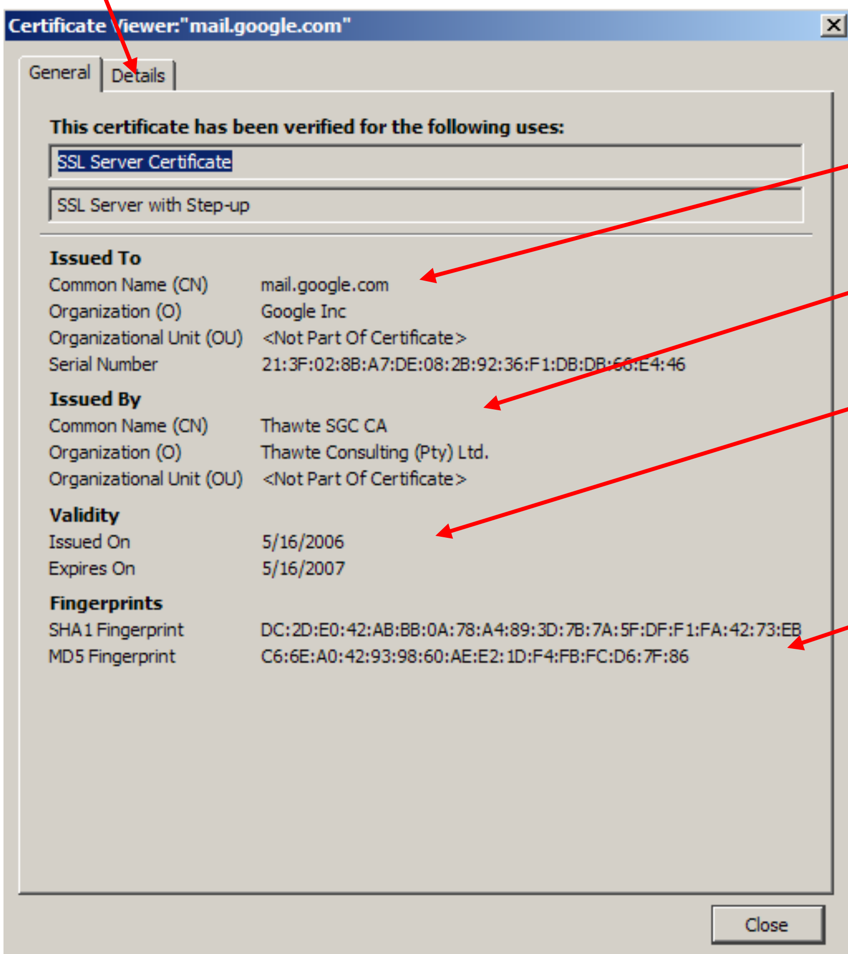
Certification Authorities

- When Alice wants Bob's public key:
  - Gets Bob's certificate (Bob or elsewhere).
  - Use CA's public key to verify the signature within Bob's certificate, then accepts public key



Certificate Contents

- info algorithm and key value itself (not shown)

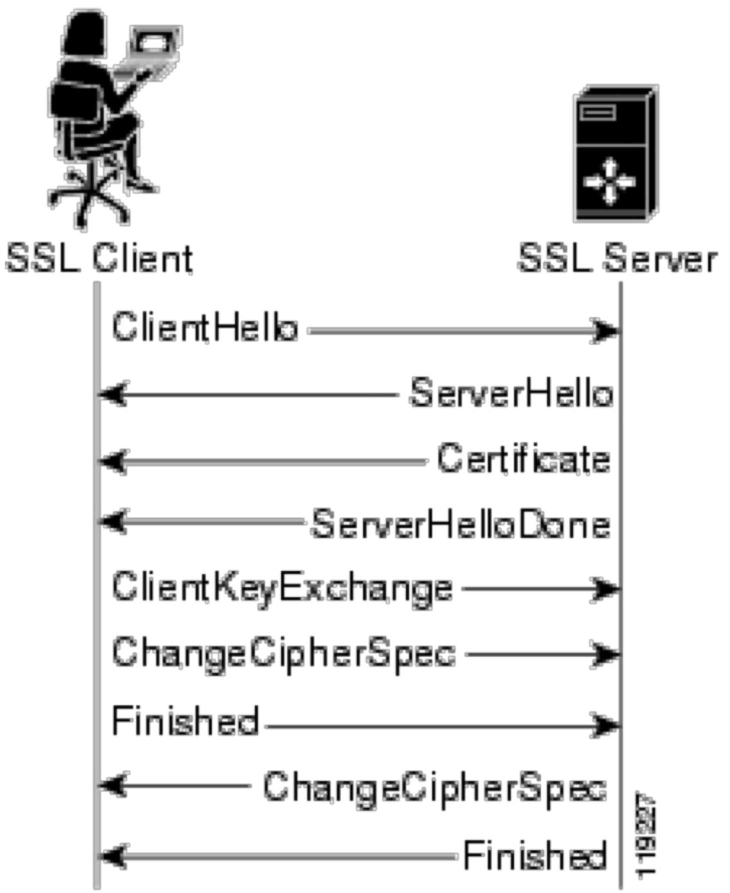


- Cert owner
- Cert issuer
- Valid dates
- Fingerprint of signature

Transport Layer Security (TLS) aka Secure Socket Layer (SSL)

- Used for protocols like HTTPS
- Special TLS socket layer between application and TCP (small changes to application).
- Handles confidentiality, integrity, and authentication.
- Uses “hybrid” cryptography.

Setup Channel with TLS "Handshake"

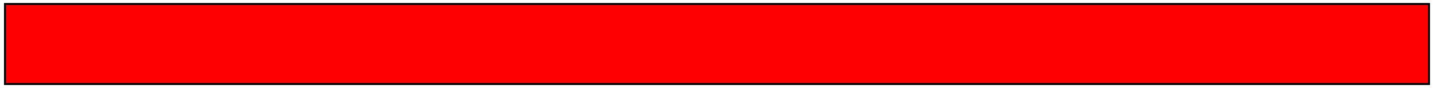


Handshake Steps:

1) Clients and servers negotiate exact cryptographic protocols
2) Client's validate public key certificate with CA public key.
3) Client encrypt secret random value with servers key, and send it as a challenge.
4) Server decrypts, proving it has the corresponding private key.
5) This value is used to derive symmetric session keys for encryption & MACs.

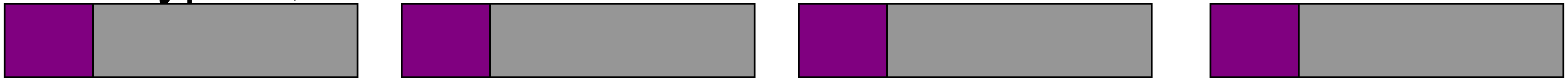
How TLS Handles Data

1) Data arrives as a stream from the application via the TLS Socket

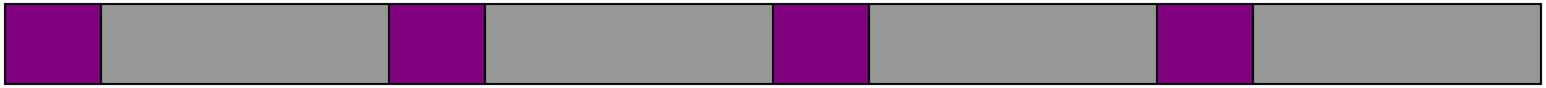


2) The data is segmented by TLS into chunks

3) A session key is used to encrypt and MAC each chunk to form a TLS “record”, which includes a short header and data that is encrypted, as well as a MAC.



4) Records form a byte stream that is fed to a TCP socket for transmission.



Analysis

- PKI lets us take the trusted third party offline:
  - If it's down, we can still talk!
  - But we trade-off ability for fast revocation
    - If server's key is compromised, we can't revoke it immediately...
    - Usual trick:
      - Certificate expires in, e.g., a year.
      - Have an on-line revocation authority that distributes a revocation list. Kinda clunky but mostly works, iff revocation is rare. Clients fetch list periodically.
- Better scaling: CA must only sign once... no matter how many connections the server handles.
- If CA is compromised, attacker can trick clients into thinking they're the real server.

Important Lessons

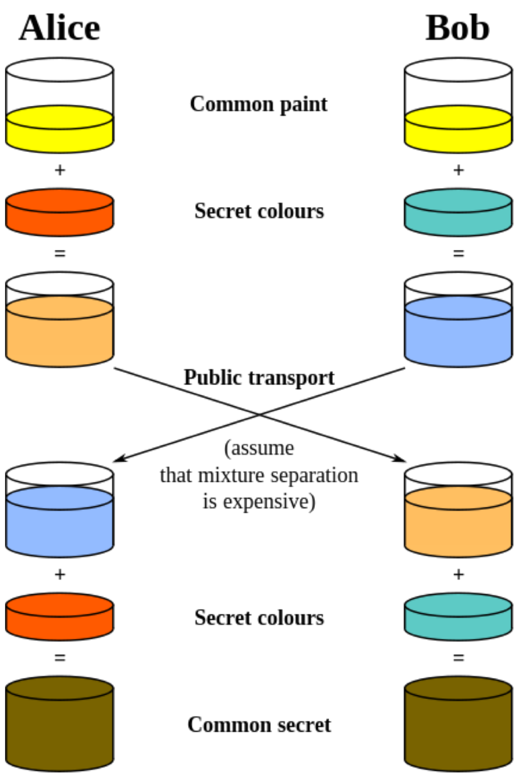
- Symmetric (pre-shared key, fast) and asymmetric (key pairs, slow) primitives provide:
  - Confidentiality
  - Integrity
  - Authentication
- "Hybrid Encryption" leverages strengths of both.
- Great complexity exists in securely acquiring keys.
- Crypto is hard to get right, so use tools from others, don't design your own (e.g. TLS).

Forward secrecy

- In KDC design, if key $K_{server\text{-}KDC}$ is compromised a year later,
  - from the traffic log, attacker can extract session key (encrypted with auth server keys).
  - attacker can decode all traffic retroactively.
- In SSL, if CA key is compromised a year later,
  - Only new traffic can be compromised. Cool…
- But in SSL, if server's key is compromised...
  - Old logged traffic can still be compromised...

Diffie-Hellman Key Exchange

- Different model of the world: How to generate keys between two people, securely, no trusted party, even if someone is listening in.



Illustrative Example
image from wikipedia
https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

Diffie-Hellman Key Exchange

- Different model of the world: How to generate keys between two people, securely, no trusted party, even if someone is listening in.

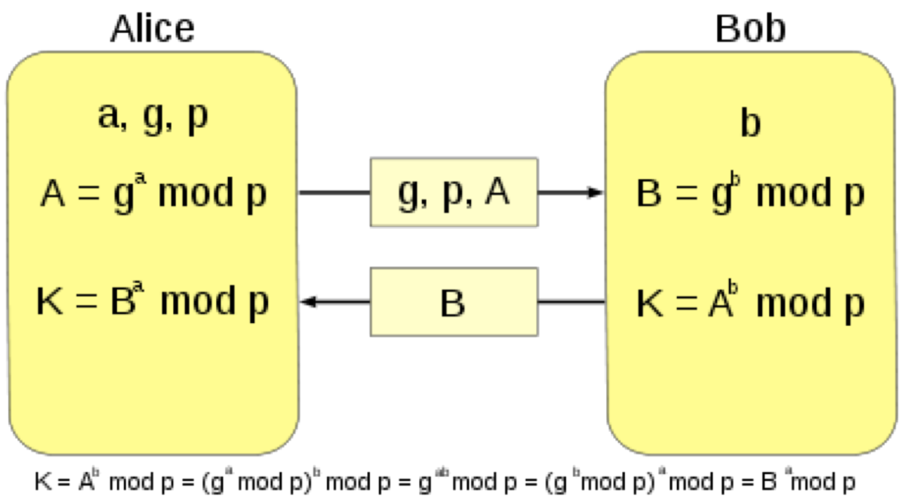


image from wikipedia

- This is cool. But: Vulnerable to man-in-the-middle attack. Attacker pair-wise negotiates keys with each of A and B and decrypts traffic in the middle. No authentication...

Authentication?

- But we already have protocols that give us authentication!
  - They just happen to be vulnerable to disclosure if long-lasting keys are compromised later...
- Hybrid solution:
  - Use diffie-hellman key exchange with the protocols we've discussed so far.
  - Auth protocols prevent M-it-M attack if keys aren't yet compromised.
  - D-H means that an attacker can't recover the real session key from a traffic log, even if they can decrypt that log.
  - Client and server discard the D-H parameters and session key after use, so can't be recovered later.
- This is called "perfect forward secrecy". Nice property.

One more note…

- public key infrastructures (PKI)s are great, but have some challenges…
  - Yesterday, we discussed how your browser trusts many, many different CAs.
  - If any one of those is compromised, an attacker can convince your browser to trust their key for a website... like your bank.
  - Often require payment, etc. (2018: LetsEncrypt)

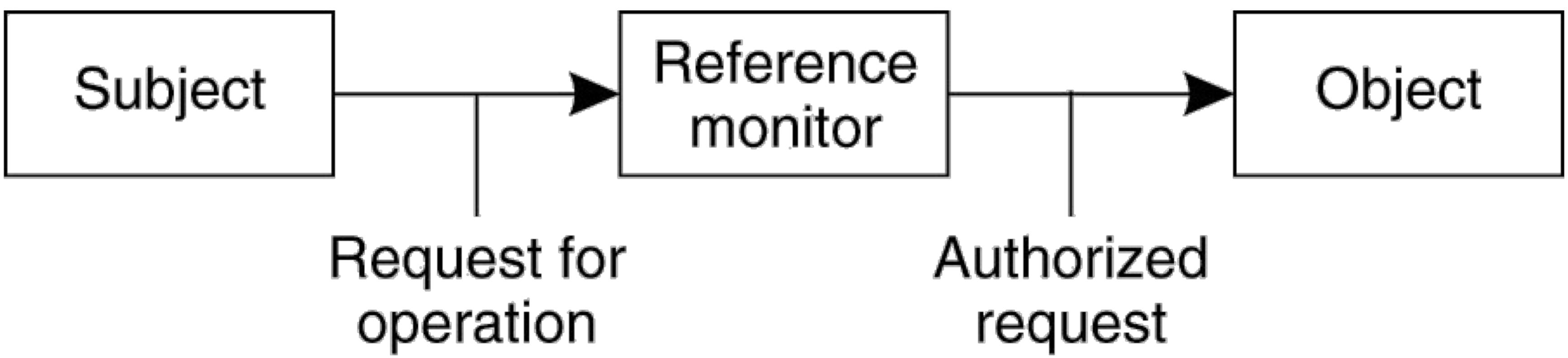
- Alternative: the "ssh" model, which we call "trust on first use" (TOFU). Sometimes called "prayer."

Today's Lecture

- Effective secure channels
- Access control
- Privacy and Tor

Access Control

- Once secure communication between a client and server has been established, we now have to worry about access control – when the client issues a request, how do we know that the client has **authorization**?



The Access Control Matrix (ACM)

A model of protection systems

- Describes who (subject) can do what (rights) to what/whom (object/subject)
- Example
  - An instructor can assign and grade homework and exams
  - A TA can grade homework
  - A Student can evaluate the instructor and TA

26

An Access Control Matrix

- Allowed Operations (Rights): r,x,w

	File1	File2	File3
Ann	rx	r	rwx
Bob	rwx	r	--
Charlie	rx	rw	w

ACMs and ACLs; Capabilities

- Real systems have to be fast and not use excessive space

What's Wrong with an ACM?

- If we have 1k 'users' and 100k 'files' and a user should only read/write his or her own files
  - The ACM will have 100k columns and 1k rows
  - Most of the 100M elements are either empty or identical
- Good for theoretical study but bad for implementation
  - Remove the empty elements?

29

Two ways to cut a table (ACM)

- Order by columns (ACL) or rows (Capability Lists)?

	File1	File2	File3
Ann	*rx*	*r*	rwx
Bob	*rwx*	r	--
Charlie	*rx*	rw	w

ACLs

Capability

Access Control Lists

- An ACL stores (non-empty elements of) each column with its object
- Columns of access control matrix

	File1	File2	File3
Andy	*rx*	*r*	rwx
Betty	*rwx*	r	--
Charlie	*rx*	rw	w

- ACLs:
- file1: { (Andy, rx) (Betty, rwx) (Charlie, rx) }
- file2: { (Andy, r) (Betty, r) (Charlie, rw) }
- file3: { (Andy, rw) (Charlie, w) }

31

Capability Lists

- Rows of access control matrix

	File1	File2	File3
Andy	*rx*	*r*	rwx
Betty	*rwx*	r	--
Charlie	*rx*	rw	w

- C-Lists:
  - Andy: { (file1, rx) (file2, r) (file3, rw) }
  - Betty: { (file1, rwx) (file2, r) }
  - Charlie: { (file1, rx) (file2, rw) (file3, w) }

32

ACLs vs. Capabilities

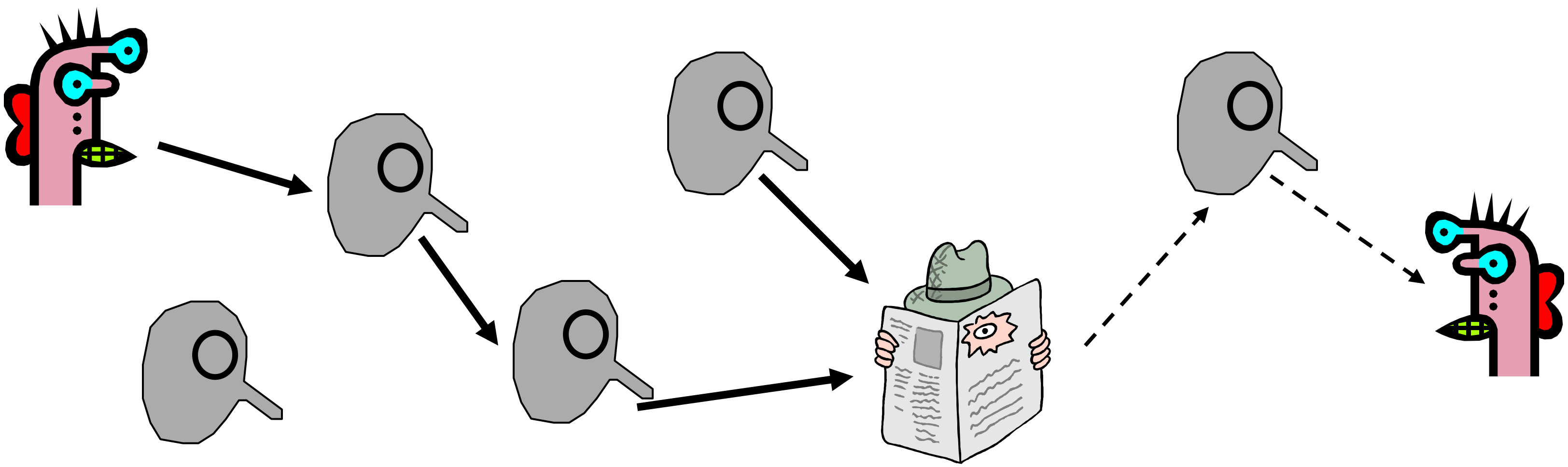
- They are equivalent:
  1. Given a subject, what objects can it access, and how?
  2. Given an object, what subjects can access it, and how?
  - ACLs answer second easily; C-Lists, answer the first easily.

- The second question in the past was most used; thus ACL-based systems are more common
- But today some operations need to answer the first question

Today's Lecture

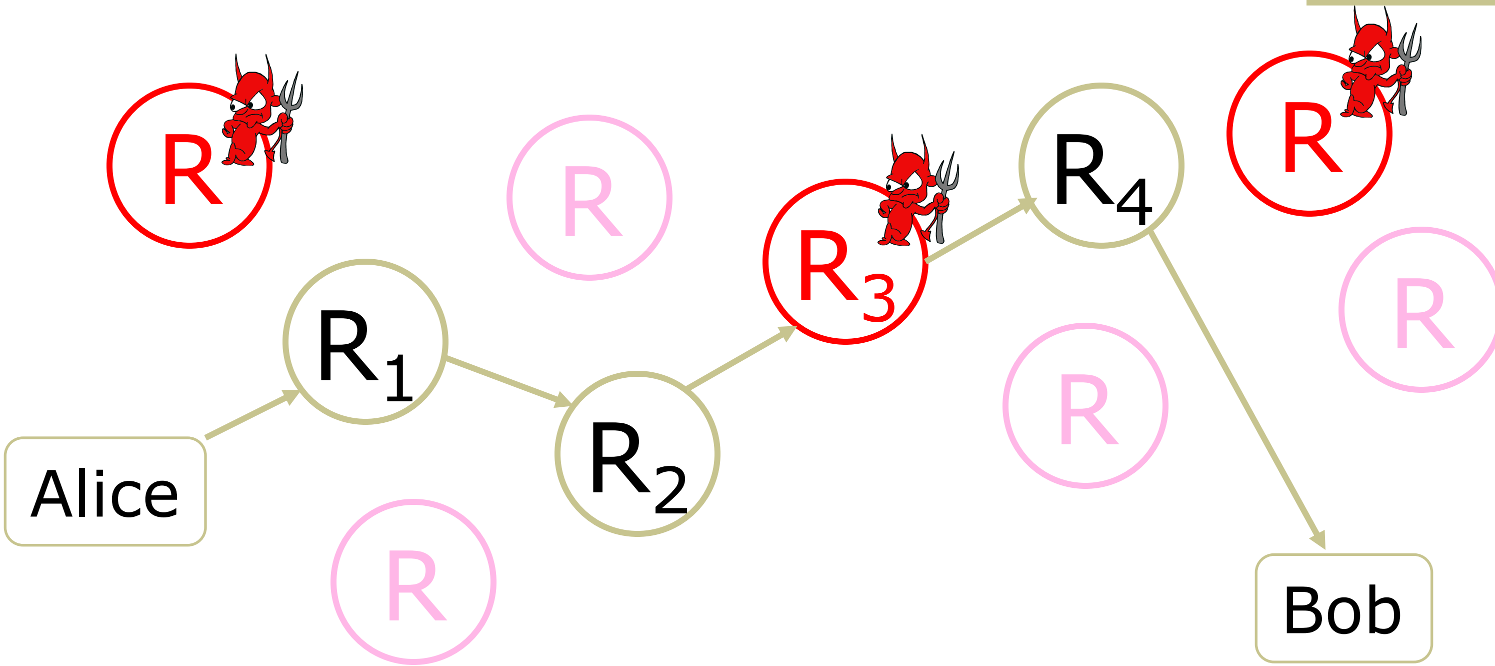
- Effective secure channels
- Access control
- Privacy and Tor
- Encryption used across the networking stack

Randomized Routing



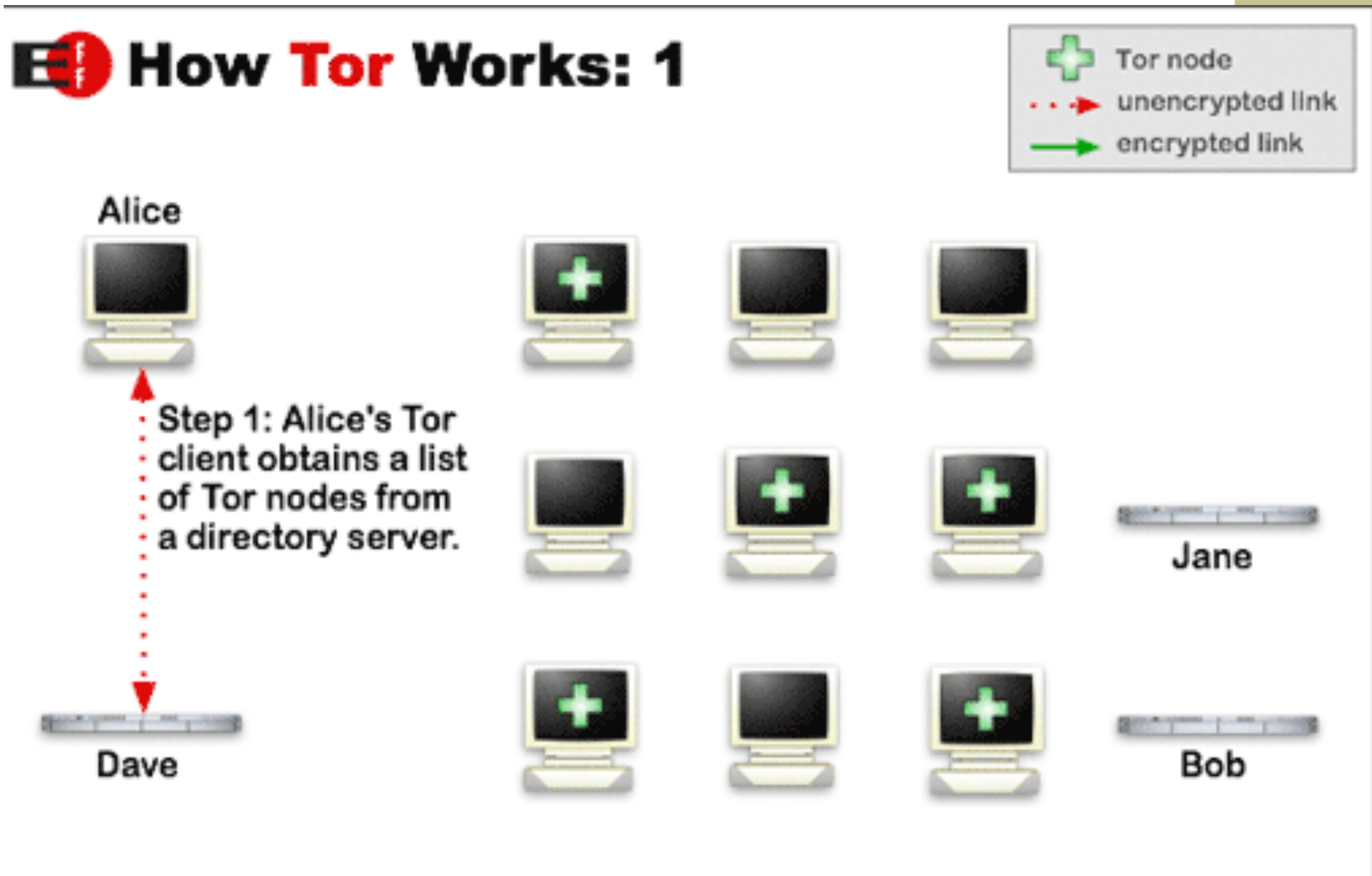
- Hide message source by routing it randomly
  - Popular technique: Crowds, Freenet, Onion routing
- Routers don't know for sure if the apparent source of a message is the true sender or another router

Onion Routing

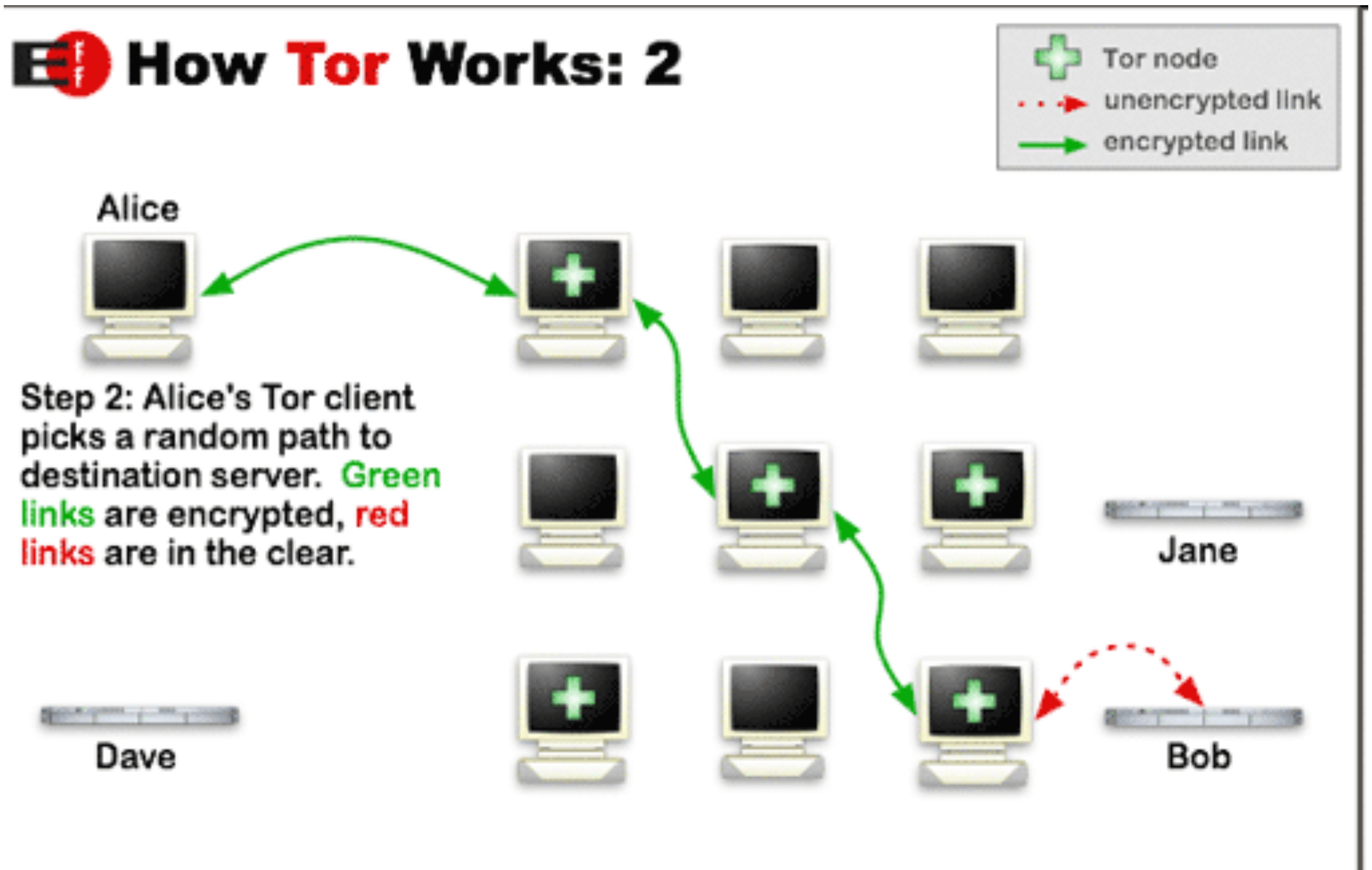


- Sender chooses a random sequence of routers
  - Some routers are honest, some controlled by attacker
  - Sender controls the length of the path

How does Tor work?

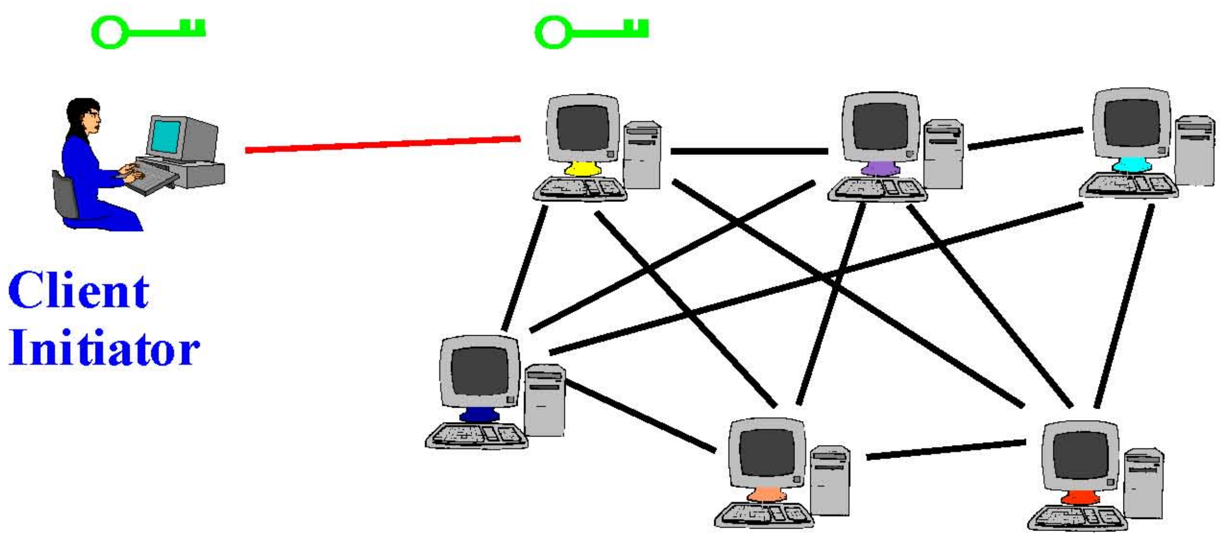


How does Tor work?



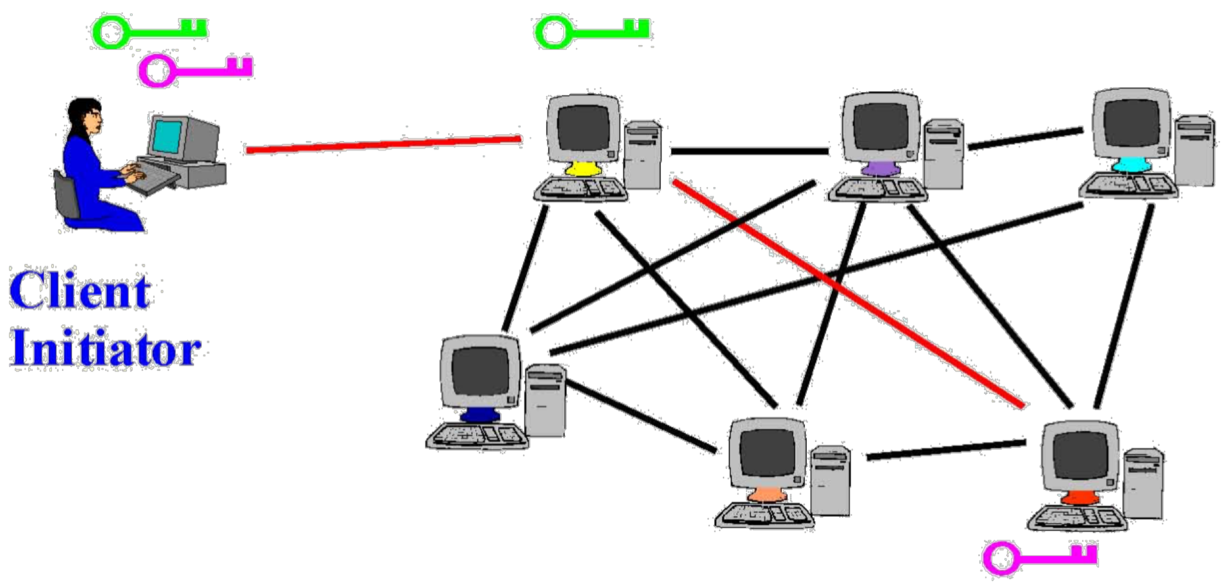
Tor Circuit Setup (1)

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



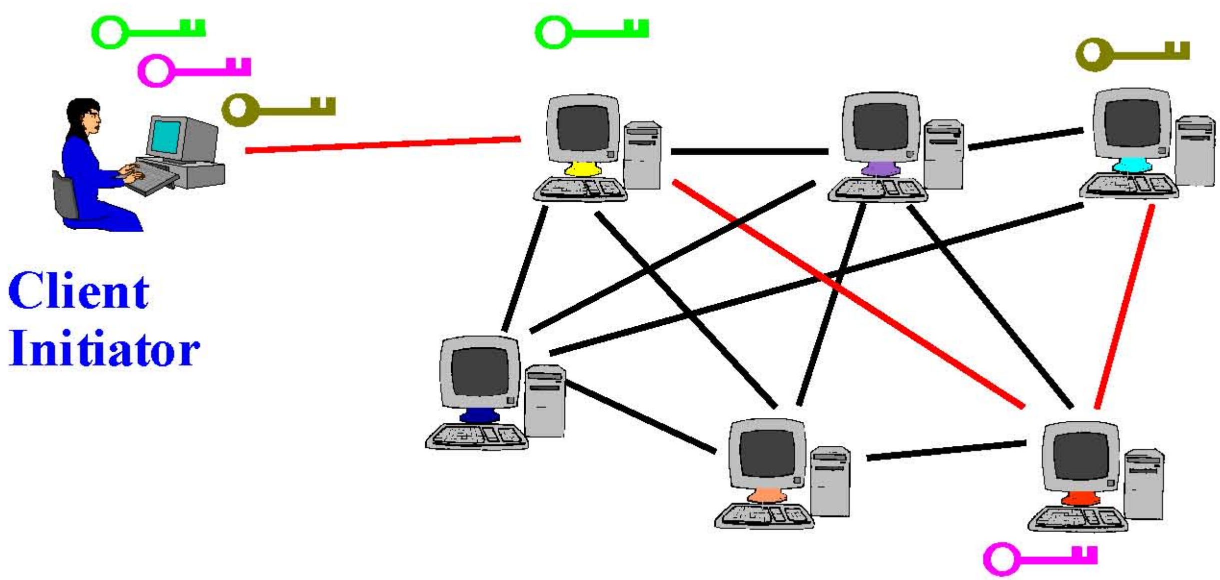
Tor Circuit Setup (2)

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

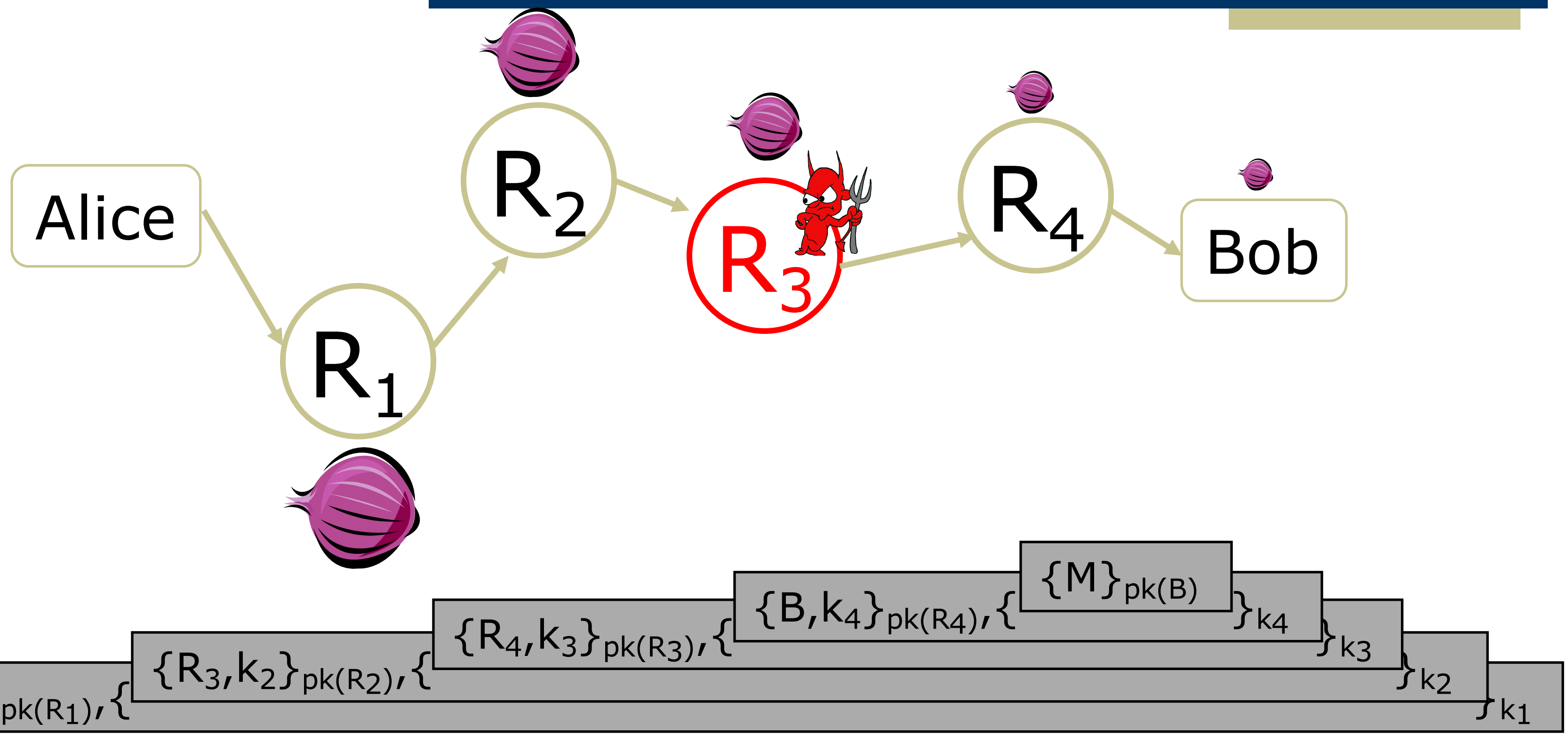


Tor Circuit Setup (3)

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



Overall Route Establishment



Routing info for each link encrypted with router's public key

Each router learns only the identity of the next router

Note: k_1, k_2, k_3 etc are session keys, so when each router (R_1, R_2, .. R_n) use their private keys to decrypt the packets, they can only then get the next hop (e.g. R_2) and the session key (k_1) to decrypt the rest of the packet and send it along.

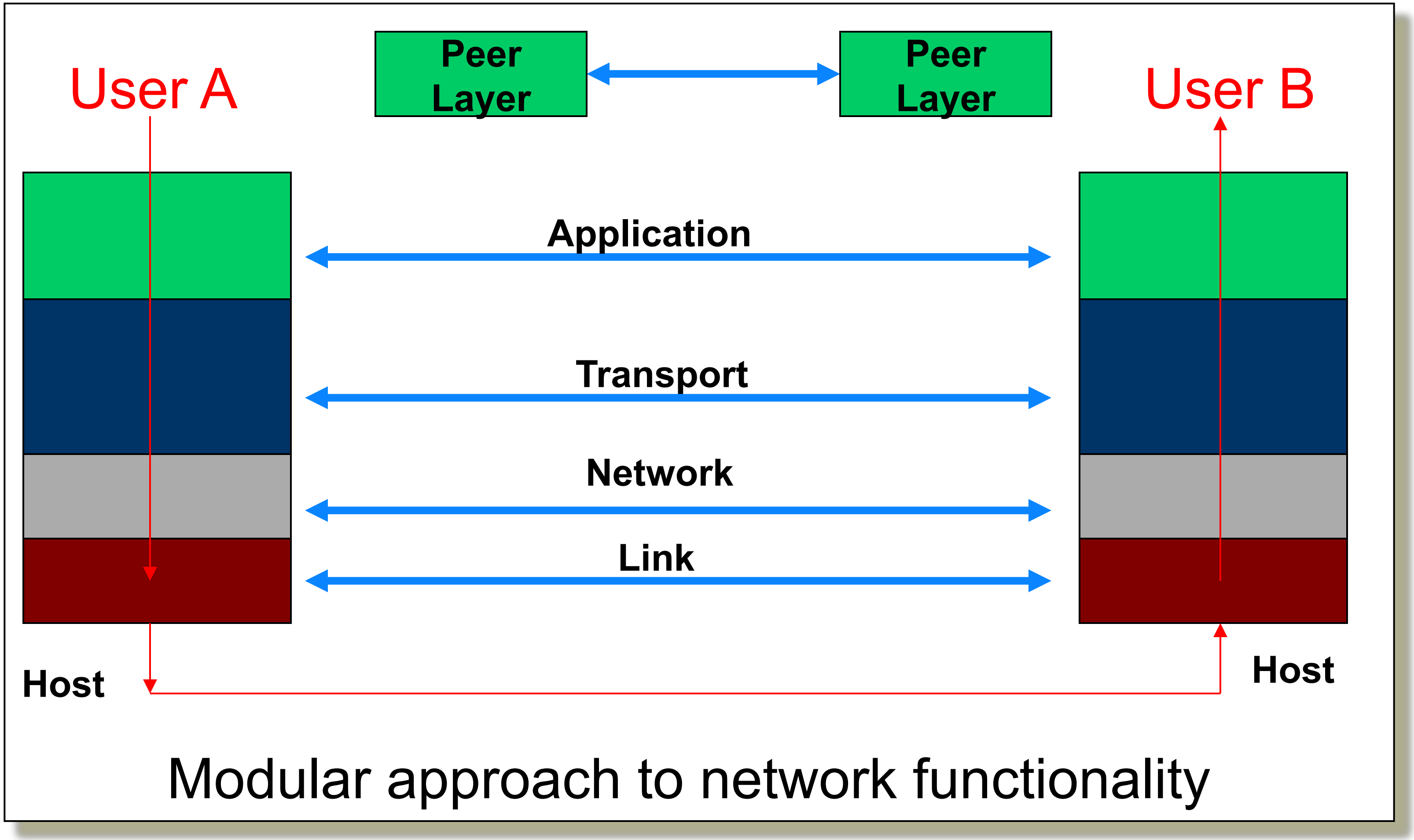
Tor

- 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, 1000s of users
- "Easy-to-use" client proxy
  - Freely available, can use it for anonymous browsing

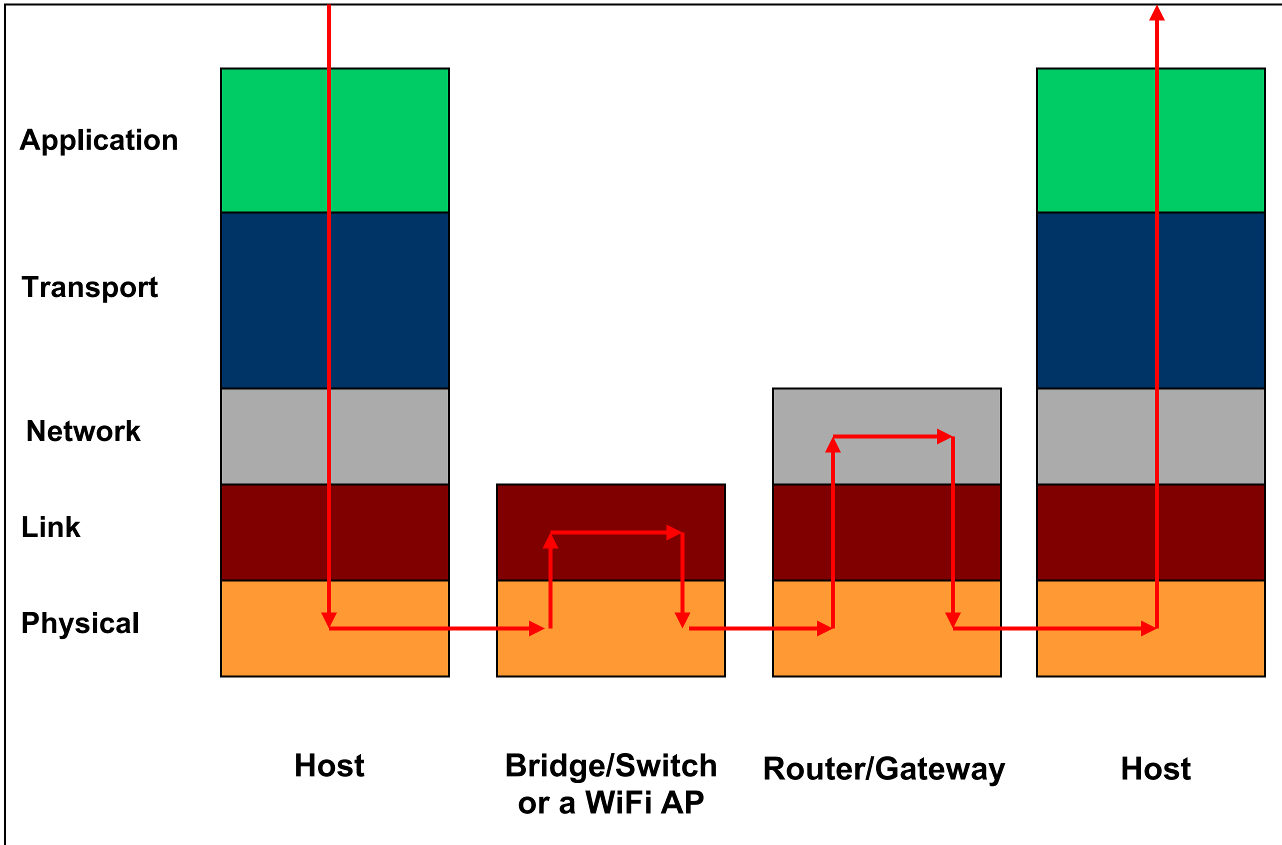
Today's Lecture

- Effective secure channels
- Access control
- Privacy and Tor
- Encryption used across the networking stack

Remember Network Layering?



IP Layering & Encryption Protocols



SSL/TLS

IPSec

802.1x, …
WPA/WEP
For WiFi

So, what does using encrypted WiFi protect against?

…. How about SSL to google.com on Starbucks open WiFi?

Key Bits: Today's Lecture

- Effective secure channels
  - Key Distribution Centers and Certificate Authorities
  - Diffie-Hellman for key establishment in the “open”
- Access control
  - Way to store what “subjects” can do to “objects”
  - Access Control Matrix: ACLs and Capability lists
- Privacy and Tor
  - Used for anonymity on the internet (Onion Routes)
  - Uses ideas from encryption, networking, P2P

Thank You!