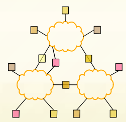
15-440 Distributed Systems Spring 2014



L-25 Security II

Today's Lecture

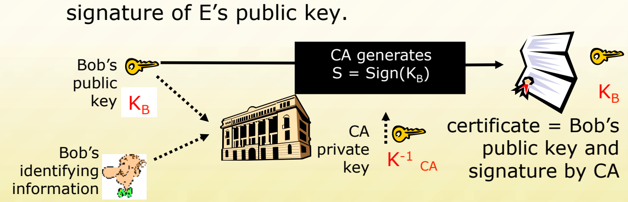
- Effective secure channels
- Access control
- Privacy and Tor

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

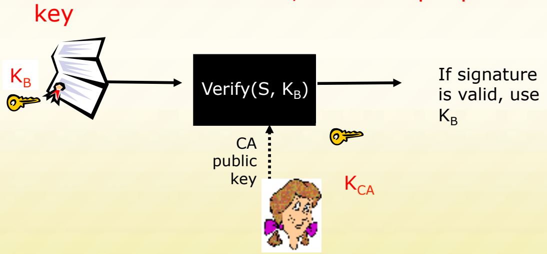
Certification Authorities

- 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.



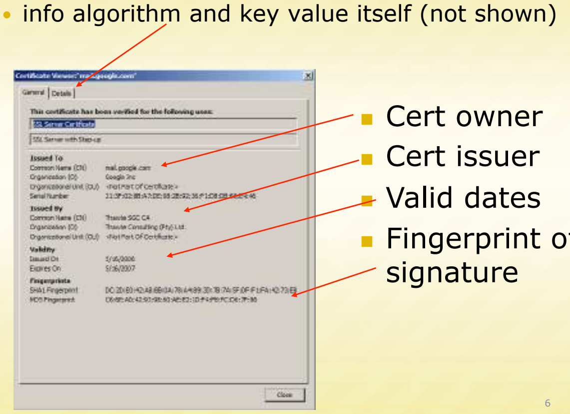
Certification Authorities

- 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

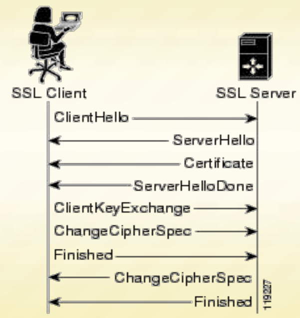
- info algorithm and key value itself (not shown)



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"



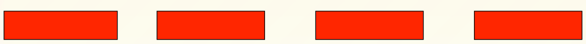
Handshake Steps:

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

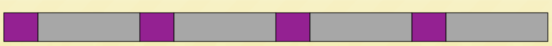
1) Data arrives as a stream from the application via the TLS Socket

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

- Public key 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

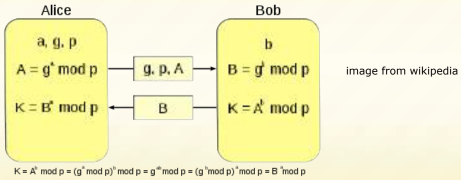
- Symmetric (pre-shared key, fast) and asymmetric (key pairs, slow) primitives provide:
  - Confidentiality
  - Integrity
  - Authentication
- "Hybrid Encryption" leverages strengths of both.
- Great complexity exists in securely acquiring keys.
- Crypto is hard to get right, so use tools from others, 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.



- 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…

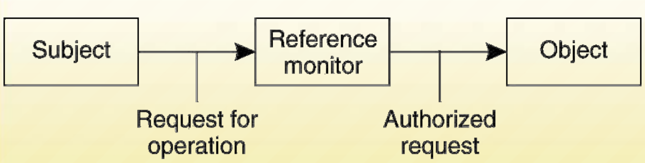
- public key infrastructures (PKI)s are great, but have some challenges…
  - Yesterday, we discussed how your browser trusts many, many different CAs.
  - If any one of those is compromised, an attacker can convince your browser to trust their key for a website... like your bank.
  - Often require payment,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

18

An Access Control Matrix

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

	File*1*	File*2*	File*3*
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?

Two ways to cut a table (ACM)

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

	File*1*	File*2*	File*3*
Ann	*rx*	*r*	rwx
Bob	*rwx*	r	--
Charlie	*rx*	rw	w

ACLs

apability

Access Control Lists

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

	File*1*	File*2*	File*3*
Andy	*rx*	*r*	rwx
Betty	*rwx*	r	--
Charlie	*rx*	rw	w

- AC
- file rx) }
- file) }
- file

Capability Lists

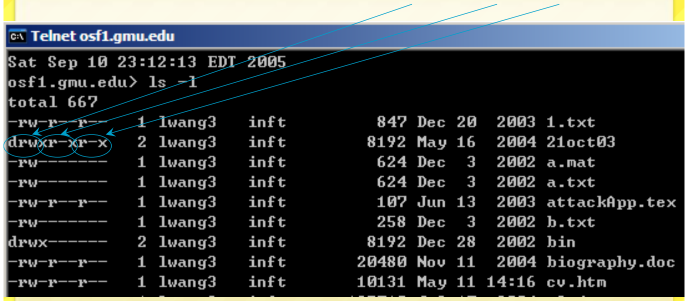
- Rows of access control matrix

	File*1*	File*2*	File*3*
Andy	*rx*	*r*	rwx
Betty	*rwx*	r	--
Charlie	*rx*	rw	w

- C-l
  - A
  - B
  - Charlie: { (file1, rx) (file2, rw) (file3, w) }

ACL:Default Permission and Abbreviation

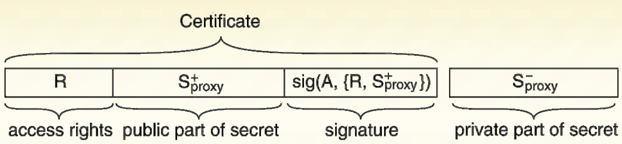
- Example: UNIX ➔
  - Three classes of users: owner, group, all others



Capability

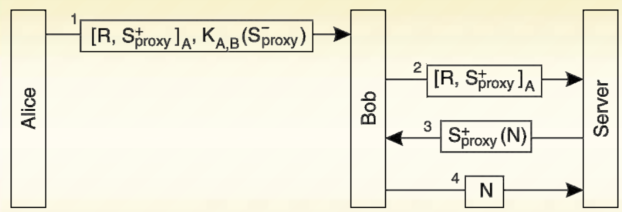
- Like a bus ticket
  - Mere possession indicates rights that subject has over object
  - Object identified by capability (as part of the token)
    - Name may be a reference, location, or something else
  - The key challenge is to prevent process/user from altering capabilities
    - Otherwise a subject can augment its capabilities at will
- Cryptography
  - Associate with each capability a cryptographic checksum enciphered using a key known to OS
  - When process presents capability, OS validates checksum

Delegation (1)



- The general structure of a proxy as used for delegation.

Delegation (2)



- Using a proxy to delegate and prove ownership of access rights.

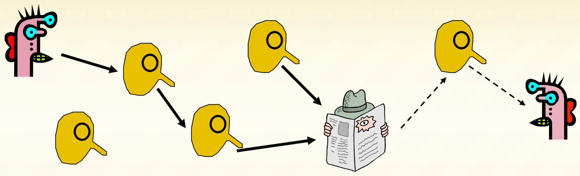
ACLs vs. Capabilities

- 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

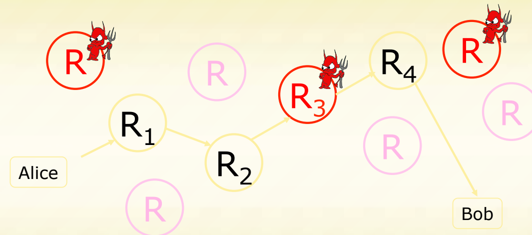
- Effective secure channels
- Access control
- Privacy and Tor

Randomized Routing



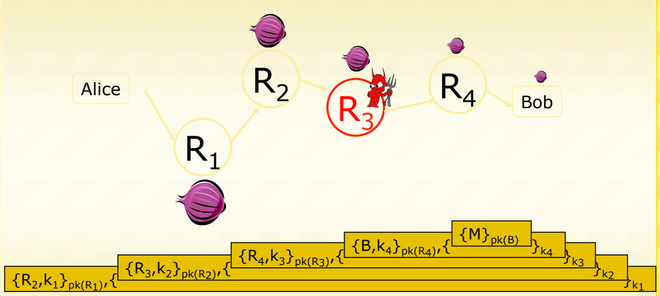
- Hide message source by routing it randomly
  - Popular technique: Crowds, Freenet, Onion routing
- Routers don't know for sure if the apparent source of a message is the true sender or another router

Onion Routing



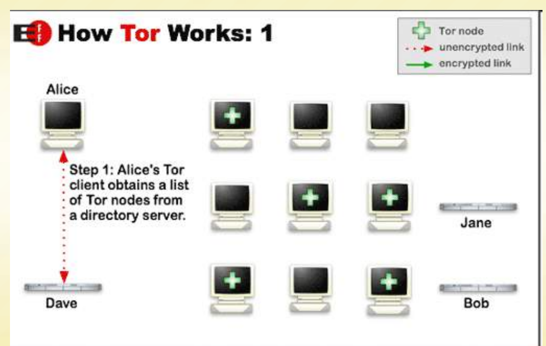
- Sender chooses a random sequence of routers
  - Some routers are honest, some controlled by attacker
  - Sender controls the length of the path

Route Establishment

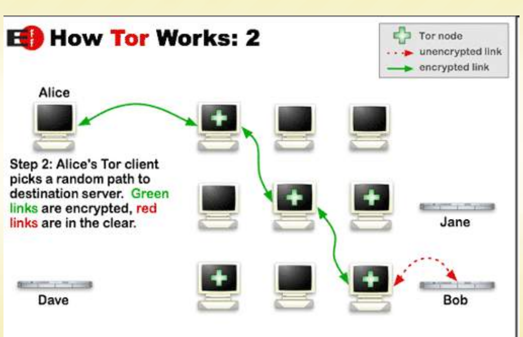


Routing info for each link encrypted with router's public key
Each router learns only the identity of the next router

How does Tor work?

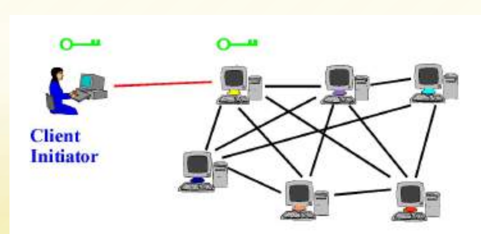


How does Tor work?



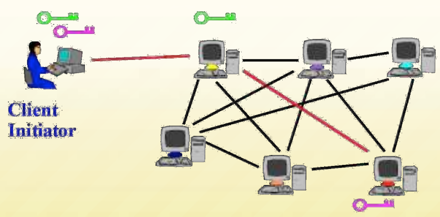
Tor Circuit Setup (1)

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



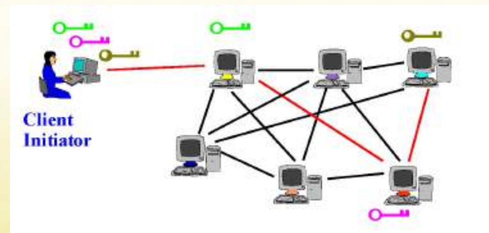
Tor Circuit Setup (2)

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



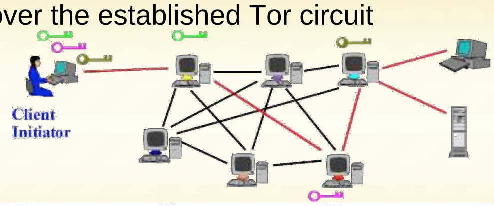
Tor Circuit Setup (3)

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



Using a Tor Circuit

- Client applications connect and communicate over the established Tor circuit



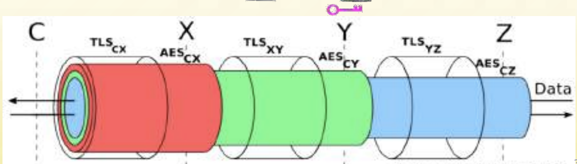
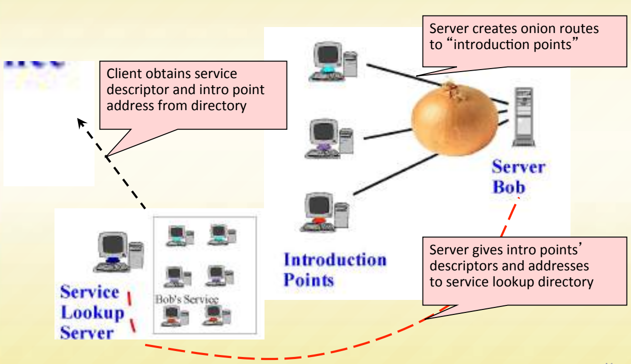


Image courtesy of Steven Murdoch

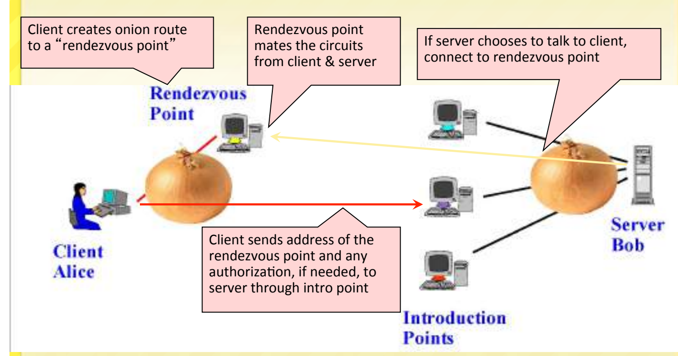
Location Hidden Servers

- 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



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

ACL Abbreviations

- Augment abbreviated lists with ACLs
  - Intent is to shorten ACL without losing the granularity
- Example → IBM AIX
  - ACL overrides base permission
  - Denial takes precedence

44

Permissions in IBM AIX

```
attributes:
base (traditional UNIX) permissions
   owner(bishop):   rw-
   group(sys):      r--
   others:     ---
extended permissions enabled
   permit    -w-    u:heidi, g=sys       [Add]
   permit    rw-    u:matt
   deny      -w-    u:holly, g=faculty   [Remove right]
```

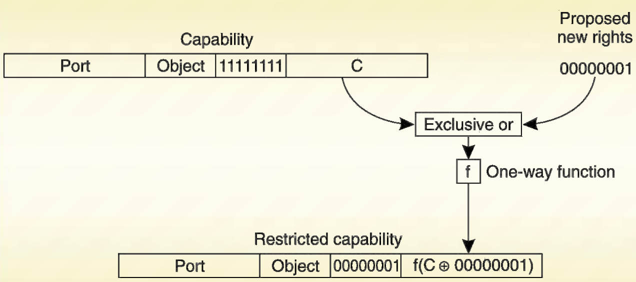

45

Capabilities and Attribute Certificates (1)

48 bits	24 bits	8 bits	48 bits
Server port	Object	Rights	Check

- Owner capability in Amoeba.

Capabilities and Attribute Certificates (2)



- Generation of a restricted capability from an owner capability.