CSE 484 (Winter 2010)

# Web Security
# +
# Asymmetric Cryptography

Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

# Goals for Today

- Web security
- Asymmetric Cryptography

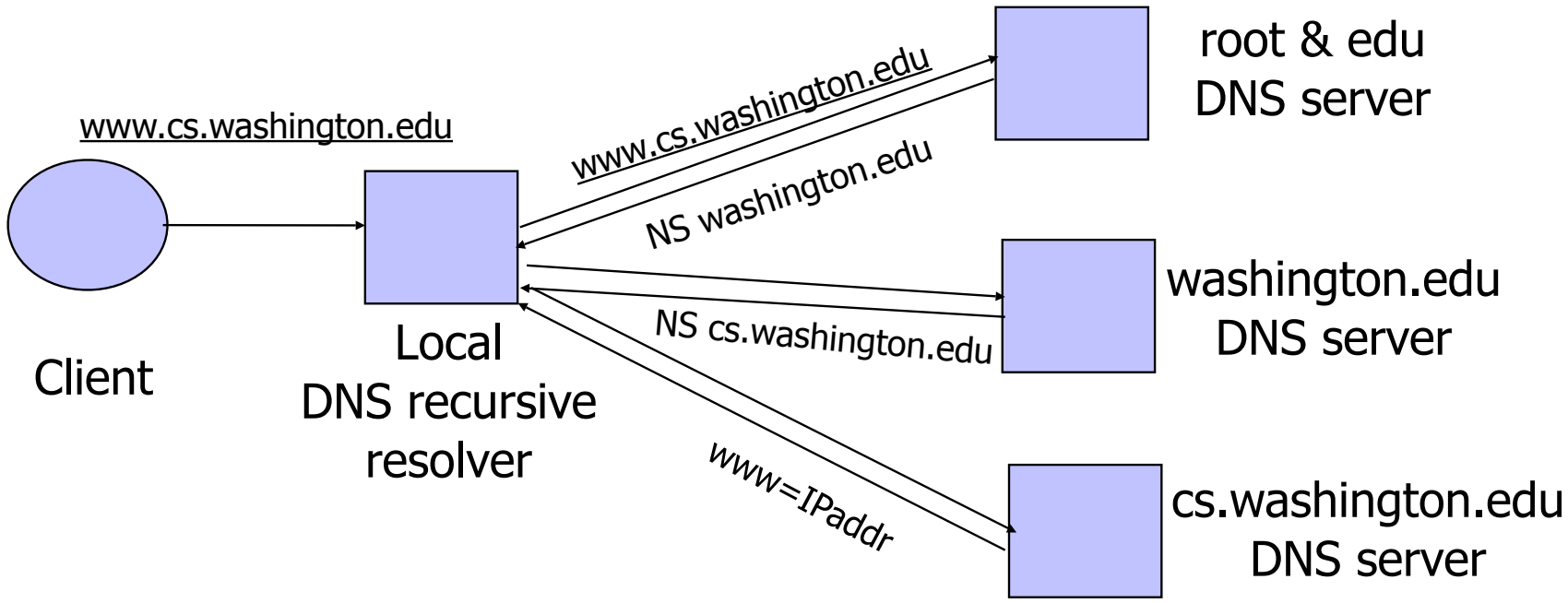
# DNS Rebinding

- JavaScript same-origin policy
  - Can only read properties of documents and windows from the same server, protocol, and port
- But can an attacker change the server?
  - Yes! If an attacker can control DNS (Domain Name Service)

# DNS: Domain Name Service

DNS maps symbolic names to numeric IP addresses

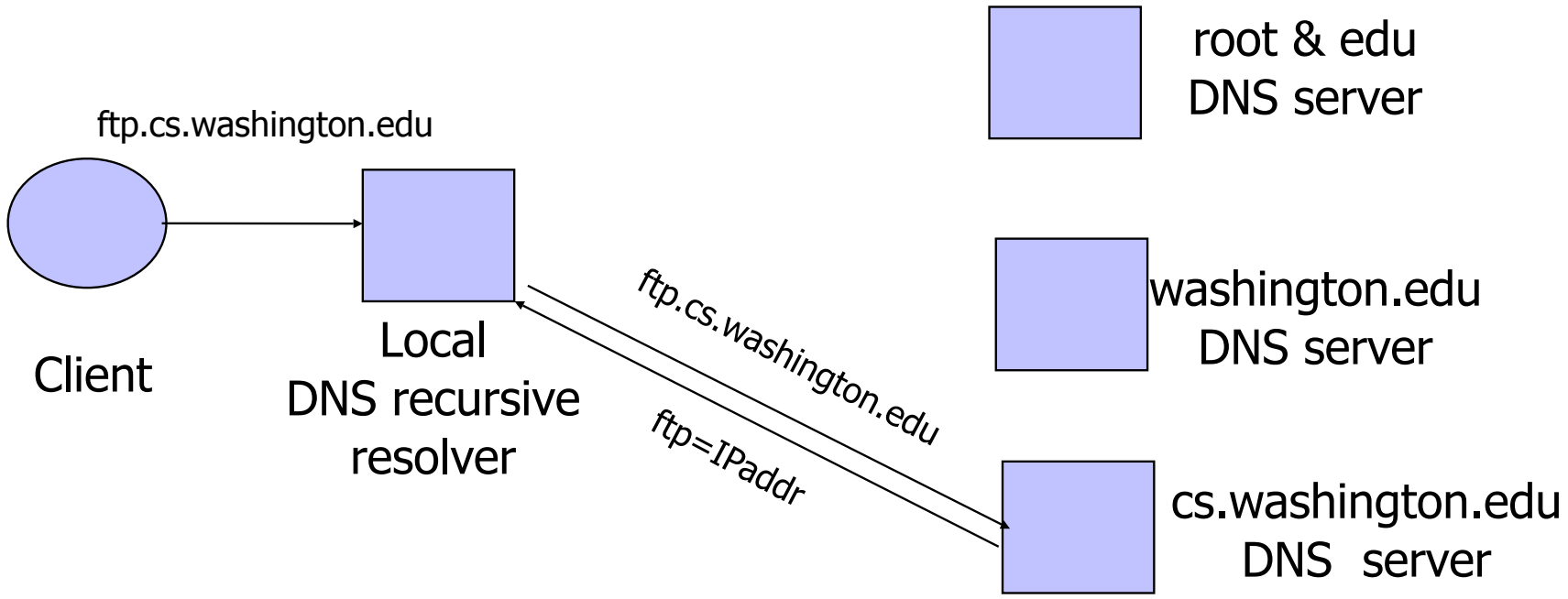
(for example, www.cs.washington.edu ↔ 128.208.3.88)



# DNS Caching

- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
    - NS records for domains
- DNS negative queries are cached
  - Don’t have to repeat past mistakes
    - For example, misspellings
- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record

# Cached Lookup Example



# DNS Vulnerabilities

- DNS host-address mappings are not authenticated
- DNS implementations have vulnerabilities
  - Reverse query buffer overrun in old releases of BIND
    - Gain root access, abort DNS service…
  - MS DNS for NT 4.0 crashes on chargen stream
    - telnet ntbox 19 | telnet ntbox 53
- Denial of service is a risk
  - If can't use DNS ... can't use the "Internet"

# Reverse DNS Spoofing

- Trusted access is often based on host names
  - E.g., permit all hosts in .rhosts to run remote shell
- Network requests such as rsh or rlogin arrive from numeric source addresses
  - System performs reverse DNS lookup to determine requester's host name and checks if it's in .rhosts
- If attacker can spoof the answer to reverse DNS query, he can fool target machine into thinking that request comes from an authorized host
  - No authentication for DNS responses and typically no double-checking (numeric → symbolic → numeric)

# Other DNS Risks

- DNS cache poisoning
  - False IP with a high time-to-live will stay in the cache of the DNS server for a long time
  - Basis of pharming
- Spoofed ICANN registration and domain hijacking
  - Authentication of domain transfers based on email addr
  - Aug ’04: teenager hijacks eBay’s German site
  - Jan ’05: hijacking of panix.com (oldest ISP in NYC)
    - "The ownership of panix.com was moved to a company in Australia, the actual DNS records were moved to a company in the United Kingdom, and Panix.com's mail has been redirected to yet another company in Canada."
- Misconfiguration and human error

# Network Solutions Under Large Scale DDoS Attack, Millions of Websites Potentially Unreachable

Jan 23, 2009 2:55 PM PST | Comments: 0 | Views: 10,429

By CircleID Reporter



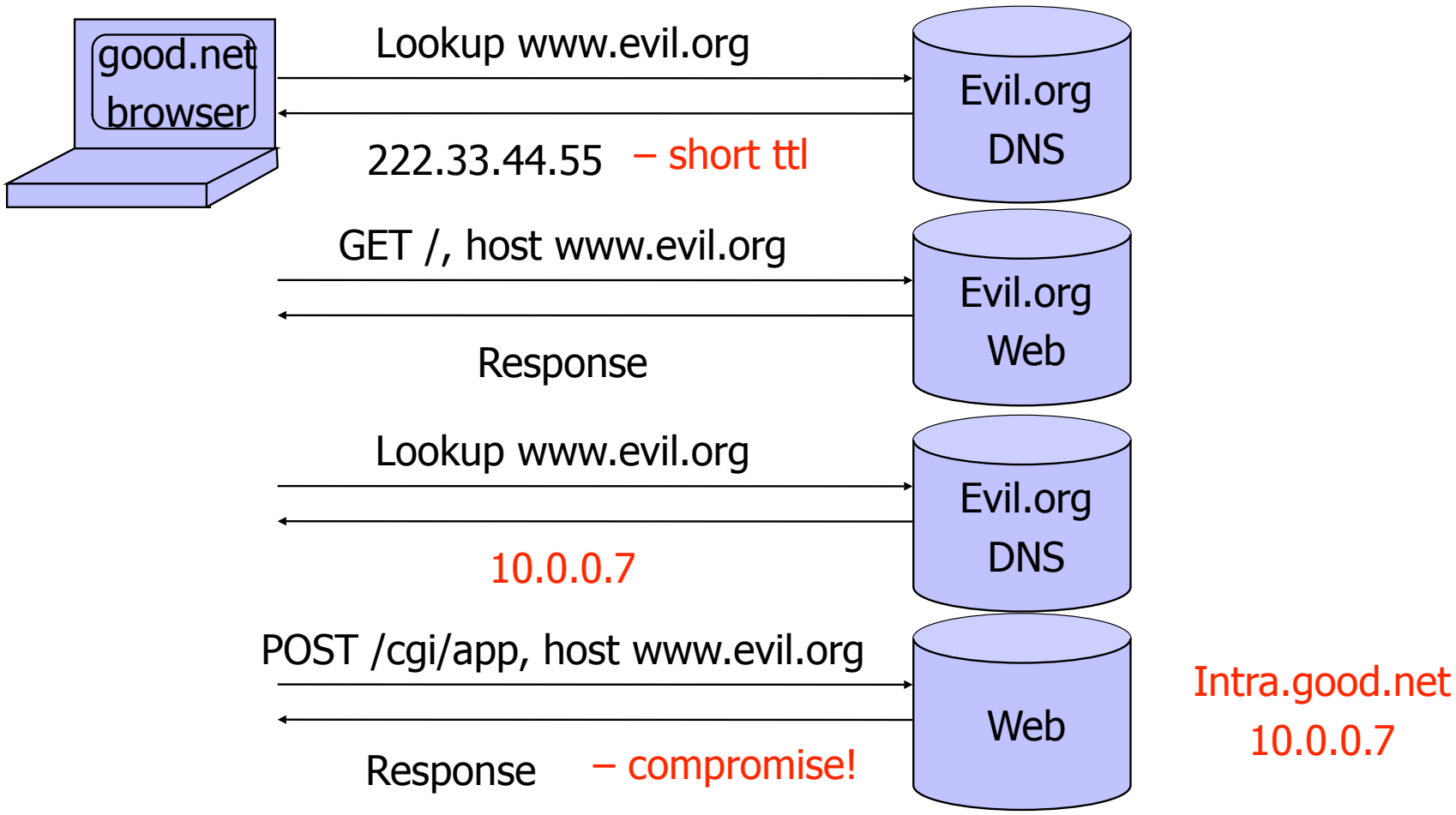
**Update Received from Network Solutions Jan 23, 2009 7:27PM PST**

"DNS queries for web sites should be responding normally. Thank you all for your understanding. As always, we will continue to work to take measures to prevent these and other types of technical issues caused by third parties that may impact our customers."

# JavaScript/DNS Intranet attack (I)

- Consider a Web server intra.good.net
  - IP: 10.0.0.7, inaccessible outside good.net network
  - Hosts sensitive CGI applications
- Attacker at evil.org gets good.net user to browse www.evil.org
- Places Javascript on www.evil.org that accesses sensitive application on intra.good.net
  - This doesn't work because Javascript is subject to "same-origin" policy
  - … but the attacker controls evil.org DNS

# JavaScript/DNS Intranet attack (II)



# General issue: Inadequate Input Validation

- http://victim.com/copy.php?name=username
- copy.php includes

  Supplied by the user!

  system("cp temp.dat $name.dat")
- User calls

  http://victim.com/copy.php?name="a; rm *"
- copy.php executes

  system("cp temp.dat a; rm *");

# User Data in SQL Queries

- set UserFound=execute(

  SELECT * FROM UserTable WHERE

  username=' " & form("user") & " ' AND

  password=' " & form("pwd") & " ' " );

  - User supplies username and password, this SQL query checks if user/password combination is in the database

- If not UserFound.EOF

  Authentication correct

  else Fail

- (Notation approximate, to focus on key issues)

Only true if the result of SQL query is not empty, i.e., user/pwd is in the database

# SQL Injection

Always true!

- User gives username ' OR 1=1 --
- Web server executes query

```
set UserFound=execute(
    SELECT * FROM UserTable WHERE
    username='' OR 1=1 -- ... );
```

Everything after -- is ignored!

- This returns the entire database!
- UserFound.EOF is always false; authentication is always "correct"

# It Gets Better (or Worse?)

- User gives username

  ’ exec cmdshell ’net user badguy badpwd’ / ADD --

- Web server executes query

  set UserFound=execute(

  SELECT * FROM UserTable WHERE

  username=’ ’ exec … -- … );

- Creates an account for badguy on DB server

# Uninitialized Inputs

```
/* php-files/lostpassword.php */
for ($i=0; $i<=7; $i++)
     $new_pass .= chr(rand(97,122))
...
$result = dbquery("UPDATE ".$db_prefix."users
    SET user_password=md5('$new_pass')
    WHERE user_id='".$data['user_id']." ' ");
```

Creates a password with 7 random characters, assuming $new_pass is set to NULL

SQL query setting password in the DB

In normal execution, this becomes

```
UPDATE users SET user_password=md5('???????')
WHERE user_id='userid'
```

# Exploit

User appends this to the URL:

&new_pass=badPwd%27%29%2c

user_level=%27103%27%2cuser_aim=%28%27

This sets $new_pass to
badPwd'), user_level='103', user_aim=('

SQL query becomes

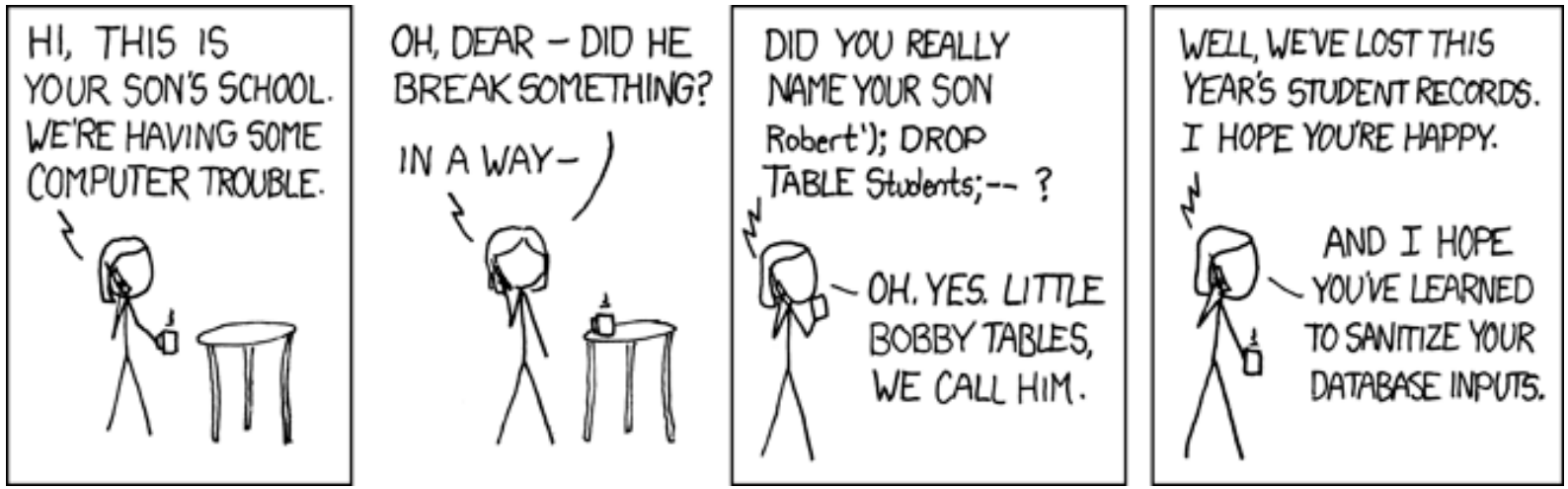
UPDATE users SET user_password=md5('badPwd')

user_level='103', user_aim=('???????')

WHERE user_id='userid'

... with superuser privileges

User's password is
set to 'badPwd'




http://xkcd.com/327/

# Dangerous Websites

- 2006 "Web patrol" study at Microsoft identified 752 unique URLs that could successfully exploit unpatched Windows XP machines
  - Many are interlinked by redirection and controlled by the same major players
- "But I never visit risky websites"
  - 11 exploit pages are among the top 10,000 most visited
  - Common trick: put up a page with popular content, get into search engines, page redirects to the exploit site
    - One of the malicious sites was providing exploits to 75 "innocuous" sites focusing on (1) celebrities, (2) song lyrics, (3) wallpapers, (4) video game cheats, and (5) wrestling
- Similar study at UW
- Now through emails and ads

## Search: Image Searchers Snared By Malware

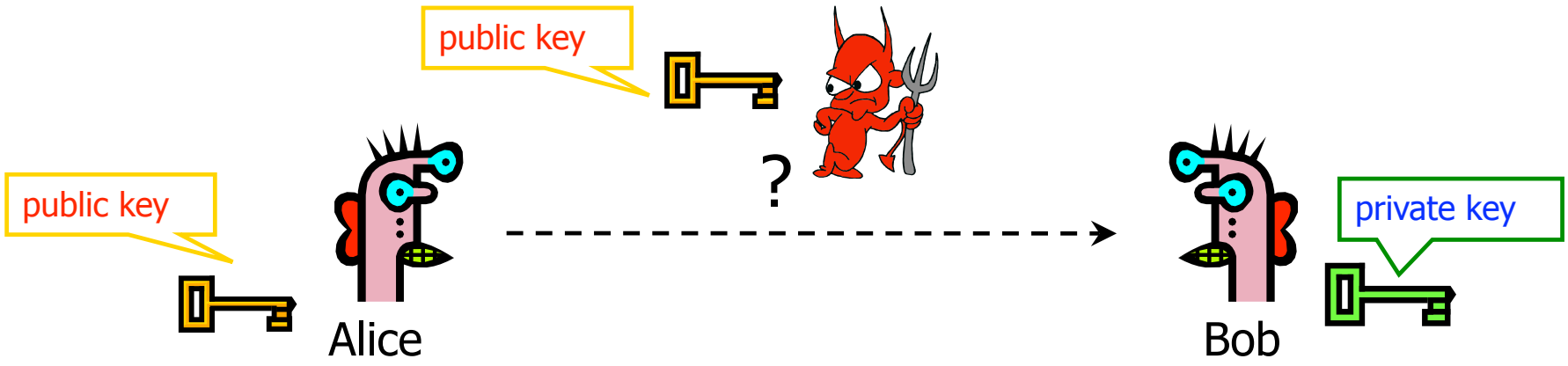
**Posted by samzenpus on Thursday February 04, @10:16AM**
**from the caught-in-the-net dept.**

Slashdot frequent contributor Bennett Haselton writes

> "Sites that have been hacked by malware writers are now serving infected content only when the visitor views the site through a frame on Google Images. This recent twist on a standard trick used by malware writers, makes it harder for webmasters and hosting companies to discover that their sites have been infected. Automated tools that check websites for infections and training procedures for hosting company abuse-department staffers will have to be updated accordingly."

# Public Key Cryptography

# Basic Problem




Given: Everybody knows Bob's public key

Only Bob knows the corresponding private key

Goals: 1. Alice wants to send a secret message to Bob

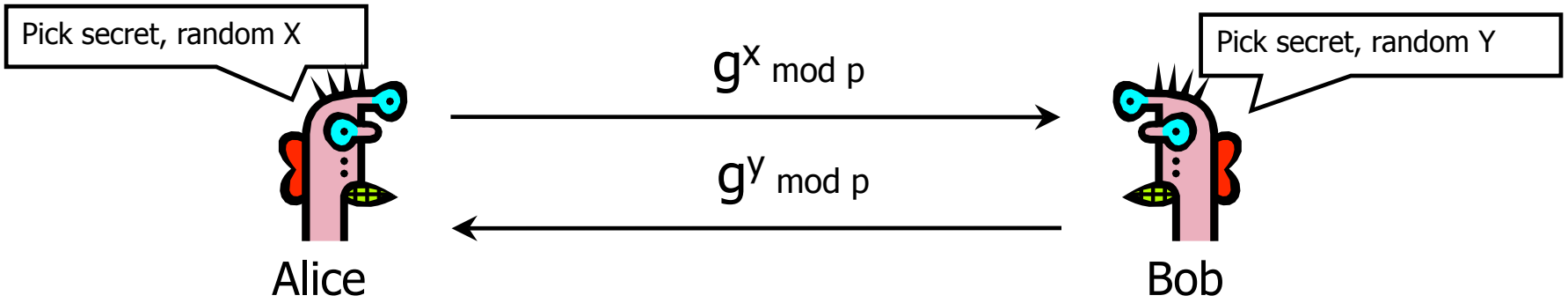
2. Bob wants to authenticate himself

# Applications of Public-Key Crypto

- Encryption for confidentiality
  - Anyone can encrypt a message
    - With symmetric crypto, must know secret key to encrypt
  - Only someone who knows private key can decrypt
  - Key management is simpler (maybe)
    - Secret is stored only at one site: good for open environments
- Digital signatures for authentication
  - Can "sign" a message with your private key
- Session key establishment
  - Exchange messages to create a secret session key
  - Then switch to symmetric cryptography (why?)

# Diffie-Hellman Protocol (1976)

- Alice and Bob never met and share no secrets
- Public info: p and g
  - p is a large prime number, g is a generator of $Z_p^*$
    - $Z_p^*=\{1, 2 \ldots p-1\}$; $\forall a \in Z_p^* \; \exists i$ such that $a=g^i \bmod p$
    - Modular arithmetic: numbers "wrap around" after they reach p




Compute $k=(g^y)^x=g^{xy} \bmod p$

Compute $k=(g^x)^y=g^{xy} \bmod p$